DLBD: A Self-Supervised Direct-Learned Binary Descriptor

Bin Xiao, Yang Hu, Bo Liu, Xiuli Bi*, Weisheng Li, Xinbo Gao
Chongqing University of Posts and Telecommunications
Chongqing, China
xiaobin@cqupt.edu.cn, S210201038@stu.cqupt.edu.cn
{boliu, bixl, liws, gaoxb}@cqupt.edu.cn

Abstract

For learning-based binary descriptors, the binarization process has not been well addressed. The reason is that the binarization blocks gradient back-propagation. Existing learning-based binary descriptors learn real-valued output, and then it is converted to binary descriptors by their proposed binarization processes. Since their binarization processes are not a component of the network, the learning-based binary descriptor cannot fully utilize the advances of deep learning. To solve this issue, we propose a model-agnostic plugin binary transformation layer (BTL), making the network directly generate binary descriptors. Then, we present the first self-supervised, direct-learned binary descriptor, dubbed DLBD. Furthermore, we propose ultra-wide temperature-scaled cross-entropy loss to adjust the distribution of learned descriptors in a larger range. Experiments demonstrate that the proposed BTL can substitute the previous binarization process. Our proposed DLBD outperforms SOTA on different tasks such as image retrieval and classification[1].

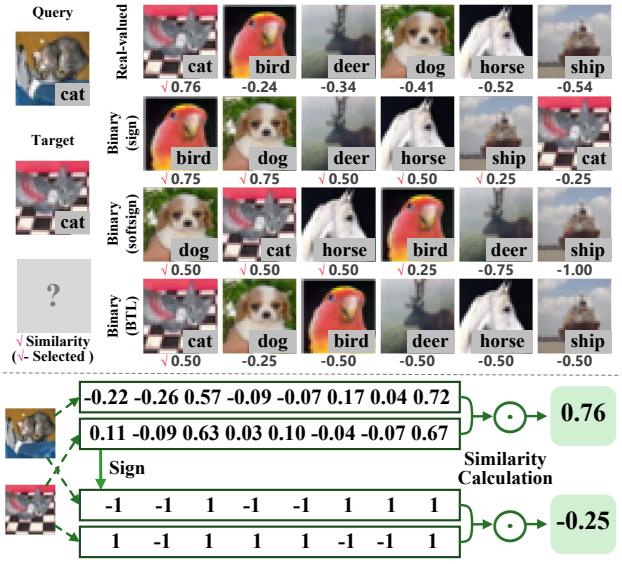


Figure 1. An example of the issue in the existing binarization processes. Images in this example come from CIFAR10, sized 32×32, and their descriptors are represented by 8-bit.

1. Introduction

Feature descriptors have played an essential role in many computer vision tasks [4, 7, 11], such as visual search, image classification, and panorama stitching. The research of descriptors is intended to bridge the gap between low-level pixels and high-level semantic information. With feature descriptors developing, lightweight binary descriptors have been proposed to further reduce memory consumption and computational complexity. The initial research focused on hand-crafted binary descriptors, such as [1, 18, 20]. Although hand-crafted binary descriptors are effective, their discriminability and robustness are eager to be improved. With the advent of Neural Networks, research has gradually moved away from hand-crafted descriptors to ones utilizing the powerful learning ability of Neural Networks.

The learning-based binary descriptors can be divided into supervised learning-based binary descriptors and unsupervised learning-based ones. The supervised learning-based binary descriptors [23, 24, 27], such as L2Net [23], have excellent performance far surpassing hand-crafted binary descriptors. However, there is a critical issue: the supervised learning-based binary descriptors depend on massive training data with human-annotated labels, which requires many labor costs. Therefore, researchers present unsupervised learning-based binary descriptors [8–10, 15, 25, 26, 29].

It is worth noting that the derivative of the binarization function as a step function at non-zero points is 0, which cannot back-propagate gradients and then prevents

*Corresponding author
[1]Our code is available at: https://github.com/CQUPT-CV/DLBD

the learning-based binary descriptors from directly training and outputting binary descriptors. For this reason, the existing learning-based binary descriptors employ separate binarization processes, which convert the real-valued output of the trained network to the binary descriptors. However, as shown in Fig. 1, the real-valued output of the network shows positive results, but the binary counterpart does not. It is because real-valued descriptors focus on the difference between values while binary descriptors only focus on the sign of values. Concretely, the 7^{th} bits of real-vallued descriptors given in Fig. 1 are 0.04 and -0.07, the difference of the bits is small but not for binary descriptors after binarization. Fig. 1 shows three binarization processes: converting the real-valued output to binary descriptor by sign, training model with softsign attached the top of model and then converting by sign ($softsign = a/(|a| + \gamma)$, where γ is a hyperparameter, selected as 0.1 by experiment), and our proposed BTL. The real-valued descriptor is also shown for comparison.

In this paper, our work can be summarized as follow:

- We proposed the Binary Transformation Layer (BTL). BTL is a model-agnostic plugin layer, yielding direct-learned binary descriptors which fully utilize the advances of deep learning.
- Inspired by SimCLR, we proposed the self-supervised learning framework to directly learn a binary descriptor (DLBD). DLBD can be deployed for various tasks, such as image retrieval, patch matching, and image classification.
- An ultra-wide temperature-scaled cross-entropy loss is proposed for DLBD, enabling more diverse distributions in representation space to satisfy various computer vision tasks.

2. RELATED WORK

Binary Descriptors. As mentioned above, the early works constructed hand-crafted binary descriptors [1, 18, 20]. For instance, [20] proposed a gray-scale invariant texture descriptor LBP. This method compares the pixel value of the central point with its circularly symmetric P neighbors in a circle of radius R and arranges the result as a binary string.

Subsequently, to achieve more effective binary descriptors, the supervised learning framework was adopted [23, 24, 27]. For instance, BinBoost [24] learns a set of nonlinear classifiers to deal with nonlinear data structure, which makes the learned binary descriptors more discriminative jointly with the boosting algorithm. L2Net [23] extracts descriptors pertinently from patches with a new progressive sampling strategy and achieves state-of-the-art results. Supervised learning-based binary descriptors utilize the powerful learning ability of the Neural Networks. However, it requires manual annotation of data. Therefore, researchers started to focus on unsupervised learning-based binary descriptors.

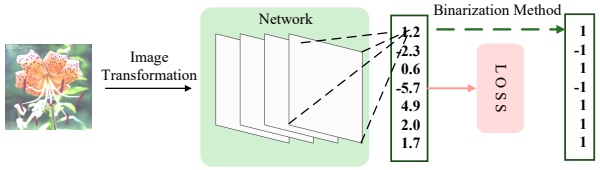
For unsupervised learning-based binary descriptors, DeepBit [15] generates compact binary descriptors under three complementary learning objectives: minimal quantization loss, evenly distributed codes, and transformation invariant bit. This method takes the original image and geometric transformed one as positive input pair and uses a Siamese Neural Network to process input paired data. During the training phase, the network's output is a real-valued descriptor. The resulting binary descriptors are obtained by a threshold. DCBD-MQ [8] adopts a K-Autoencoders (KAEs) network to obtain binary descriptors with a fine-grained multi-quantization and learn optimal allocation of bits with the fixed binary length in a competitive manner, where informative dimensions gain more bits. BLCD [10] trains a modified L2Net network for pairwise patches through locality consistency and self-distinctiveness and produces binary codes by directly quantizing the real-valued outputs. It adopts special batch normalization instead of additional loss functions to minimize the quantization error.

Self-Supervised Learning. Compared to unsupervised learning, self-supervised learning still needs supervision, but that is not manually annotated. Recently, self-supervised learning realizes powerful representations with instance discrimination which treats each image in a training set as a single class and aims to learn a feature representation that discriminates among all the classes. Visual representation methods based on self-supervised learning [5, 6, 12–14] show excellent performance and potential for self-supervision and have outperformed supervised pre-training on many downstream tasks. Chen et al. [6] present a simple framework for contrastive learning of visual representations, dubbed SimCLR. SimCLR improves the quality of learned representations through a suitable composition of data augmentations, a nonlinear projection head, and a wider Neural Network.

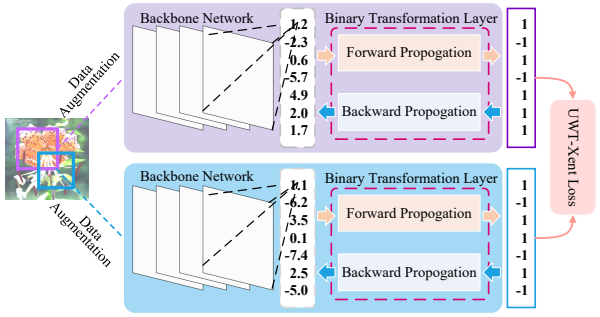
3. A Self-Supervised Direct-Learned Binary Descriptor (DLBD)

When the networks learn the real-valued descriptors, the networks are optimized by distance or similarity between real-valued descriptors. However, the corresponding binary descriptors only focus on the sign of values, which are converted from real-valued ones. This binarization process will cause the converted binary descriptors cannot accurately inherit the discriminability of the real-valued descriptors, as shown in Fig. 1.

We therefore propose BTL to enable models to learn discriminative binary descriptors directly. Besides, inspired by SimCLR, we propose the self-supervised learning frame-



(a) The existing learning-based framework for binary descriptors



(b) The proposed self-supervised learning framework directly learns binary descriptors

Figure 2. Comparing our proposed self-supervised learning framework with the previous works. (a) The existing learning-based framework for binary descriptors: the binary descriptor is not directly learned by the network. (b) The proposed self-supervised learning framework directly learns binary descriptors by inserting plugin layer, BTL, which participates in back-propagation.

work to learn binary descriptors by inserting BTL directly. Fig. 2 compares the proposed DLBD with the previous works. The proposed self-supervised learning framework consists of two Neural Network with shared parameters. The networks take as input the two different augmented views of each image in training dataset. The two augmented views of an image serve as a positive pair, and one of the augmented views with the augmented views of the other images serve as negative pairs when computing loss. After training, the output of BTL is the final direct-learned binary descriptor.

3.1. Binary Transformation Layer

Since the derivative of the binarization function as a step function at non-zero points is 0, which blocks the back-propagation of gradient-based optimization algorithm, preventing the learning-based binary descriptors from directly learning binary descriptors. Among the previous methods, BinGan [29] aims to bridge the gap between the learned real-valued descriptors and the converted binary descriptors by the softsign function; DeepBit [15] used two of three loss functions to achieve the same purpose but increases the difficulty of convergence; BLCD [10] adopted special batch normalization for sign function. Although these specific binarization processes work well, they did not solve the essential issue: how to learn discriminative binary descriptors by Neural Network directly.

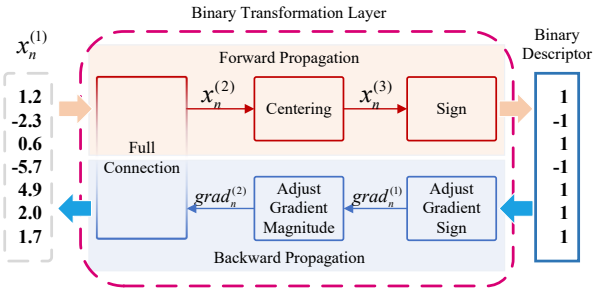


Figure 3. An illustration of the proposed model-agnostic BTL, which makes binarization function work in the back-propagation algorithm.

To break this bottleneck, we design BTL, as shown in Fig. 3. BTL aligns well with the optimization for binary descriptors, since the calculation of the objective function is based on binary descriptors instead of real-valued ones. And then, we need to guarantee that the feedback is powerful and drives network optimization in the right direction due to the gap between real-valued and binary descriptors.

In forward propagation, a fully connected layer changes the dimension of output to suit our needs. The centering limits the output distribution to both sides of zero, and then converts it to the binary output. Therefore, the forward propagation in BTL can be described as:

- The k^{th} bit of the output $x_n^{(2)}$ of the fully connected layer can be calculated by

$$x_{nk}^{(2)} = f_k(x_n^{(1)}; W_k), \tag{1}$$

where $f_k(\cdot)$ represents the projection function for the k^{th} bit of the output $x_n^{(2)}$, and W_k is the parameter set of the projection function f_k. The subscript n of $x_n^{(1)}$ indicates the index of the input image.

- The output $x_n^{(2)}$ will be calculated by

$$x_{nk}^{(3)} = x_{nk}^{(2)} - \mu_n. \tag{2}$$

In this formula, μ_n is the mean of all the bit in the vector $x_n^{(2)}$. By Eq. (2), the input $x_n^{(2)}$ will be centered as $x_n^{(3)}$.

- $x_{nk}^{(3)}$ is processed by

$$b_{nk} = sign(x_{nk}^{(3)}), \tag{3}$$

which compares the value $x_n^{(3)}$ with the threshold of 0 and arranges the result to -1 or 1. Then, we can get the binary vector $b_n = [b_{n1}, b_{n2}, ..., b_{nK}] \in \{-1, 1\}^{1\times K}$.

The reason for using -1 and 1 as the binary values is that the ℓ_2 norm of binary descriptors is constant and the symmetry of magnitude is usually in favor of optimization. We will take these advantages to simplify the similarity calculation and enable the BTL.

In backward propagation, we adjust suitable signs and magnitudes of gradients separately and follow the original design for the fully connected layer. Adjusting suitable signs of gradients is for the right direction of optimization, while adjusting the magnitudes of gradients is for the high efficiency of optimization. We focus on the difference between the magnitudes of gradients of $x_n^{(3)}$, since the overall magnitudes of gradients can be tuned by learning rate. It is worth mentioning that the binary values taking the same magnitude (e.g. $\{-1, 1\}$) are important to keep the balance of the optimization intensity.

- The sign of gradients can be adjusted by

$$grad_{nk}^{(1)} = \frac{\partial \text{loss}}{\partial b_{nk}}, \tag{4}$$

  where n is the index of the input image and b_n represents the corresponding binary descriptor.

- The focus of this part is to fully optimizate each bit of the descriptor by adjusting the magnitudes of gradients. The formula is as follows

$$\begin{aligned} grad_{nk}^{(2)} &= \frac{\partial S(x_n^{(3)})}{\partial x_{nk}^{(3)}} \cdot grad_n^{(1)} \\ &= (1 - \frac{(x_{nk}^{(3)})^2}{\left\|x_n^{(3)}\right\|_2^2}) \frac{grad_{nk}^{(1)}}{\left\|x_n^{(3)}\right\|_2} - \sum_{d \neq k} \frac{x_{nk}^{(3)} \cdot x_{nd}^{(3)}}{\left\|x_n^{(3)}\right\|_2^2} \frac{grad_{nd}^{(1)}}{\left\|x_n^{(3)}\right\|_2} \end{aligned}, \tag{5}$$

  where $S(x_n^{(3)})$ can be writen as

$$S\left(x_n^{(3)}\right) = \frac{x_n^{(3)}}{\left\|x_n^{(3)}\right\|_2}. \tag{6}$$

  In this formula, $\|*\|_2$ is ℓ_2-normlization.

- As for the fully connected layer, it works well in backpropagation, and we do nothing else.

The design of the BTL is shown above, and the principles of the design are described in the following. Constrastive loss (The specific ones can be seen in Eq. (12) and Eq. (13).) can be written as

$$\ell(i,j) = -\log \frac{g(sim(b_i, b_j))}{\sum_{\mathrm{m} \neq i} g(sim(b_i, b_m))}, \tag{7}$$

where sim represents a similarity calculation and g represents a transformation function. (b_i, b_j) is the positive pair of descriptors and the others are negative. We can transform Eq. (7) as follows:

$$\begin{aligned} \ell(i,j) &= \log \frac{\sum_{\mathrm{m} \neq i} g(sim(b_i, b_m))}{g(sim(b_i, b_j))} \\ &= \log \left\{ 1 + \frac{\sum_{\mathrm{m} \neq i,j} g(sim(b_i, b_m))}{g(sim(b_i, b_j))} \right\} \end{aligned}. \tag{8}$$

After transforming, the positive pair is decoupled from the negative pairs in Eq. (8), and then we can deduce the optimization trend for $x_{ik}^{(3)}$ (i.e. the sign of the gradient). Firstly, consider the influence of the negative pairs on $x_{ik}^{(3)}$,

$$\begin{aligned} \frac{\partial \ell}{\partial sim_{neg}} \frac{\partial sim_{neg}}{\partial b_{ik}} &= \left| \frac{\partial \ell}{\partial sim_{neg}} \right| \frac{\partial sim_{neg}}{\partial b_{ik}} \\ &= \begin{cases} \left| \frac{\partial \ell}{\partial sim_{neg}} \right| > 0, b_{mk} = 1 \\ -\left| \frac{\partial \ell}{\partial sim_{neg}} \right| < 0, b_{mk} = -1 \end{cases} \end{aligned}, \tag{9}$$

where sim_{neg} represents $sim(b_i, b_m))$ while $\mathrm{m} \neq \mathrm{i}, \mathrm{j}$. So far, we have calculated the sign of the gradient. Given the mechanism of Stochastic Gradient Descent (SGD) as optimization, we know the optimization trend of $x_{ik}^{(3)}$. Consider $b_{ik} \in \{-1, 1\}$ and Eq. (4), there are only 4 cases:

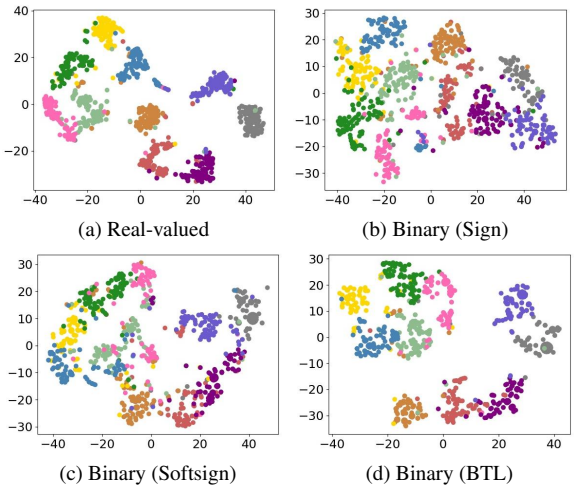
$$\text{The value of } x_{ik}^{(3)} \begin{cases} \downarrow, (b_{ik}, b_{mk}) = (1, 1) \\ \downarrow, (b_{ik}, b_{mk}) = (-1, 1) \\ \uparrow, (b_{ik}, b_{mk}) = (1, -1) \\ \uparrow, (b_{ik}, b_{mk}) = (-1, -1) \end{cases}. \tag{10}$$

These are our desired results, and for the deduction of the positive pair, the results would still be as expected. As shown in Eq. (10), increasing the magnitudes of $x_{ik}^{(3)}$ while b_{ik} and b_{mk} are different and decreasing the magnitudes of $x_{ik}^{(3)}$ while they are same. Therefore, the sign of gradients can be adjusted by Eq. (4).

As for Eq. (5), we aim to help the network learn the efficent binary descriptors by adjusting the magnitude of gradient, rather than estimating gradient precisely. As mentioned above, we focus on the difference between the magnitudes of gradients of different bits and iterations, while the learning rate can change the overall magnitudes of gradients. Concretely, given an example based on SGD mechanism but without Eq. (5), the updated $x_{nk}^{(3)}$ after one iteration is

$$x_{nk}^{(3)(t+1)} = x_{nk}^{(3)(t)} - lr \cdot \frac{\partial \ell}{\partial b_{nk}}, \tag{11}$$

where lr is the learning rate and t represents the t^{th} iteration. Eq. (11) shows that whatever the $x_{nk}^{(3)(t)}$ is 0.001 or 1,000, the changing amounts ($lr \cdot \frac{\partial \ell}{\partial b_{nk}}$ in Eq. (11)) are unchanged in both situations. Therefore, Eq. (5) constrains the magnitude of $x_{nk}^{(3)}$ to be from 0 to 1, which ensures that $grad_{nk}^{(1)}$ is powerful for real values. Furthermore, as a scaling, the input and output of Eq. (5) are positively correlated, and the output has the same sign as the input. This guarantees that the magnitude of $x_{nk}^{(3)}$ reflects the accumulation of gradients and the bits closer to 0 in the descriptor get updated first. Eq. (5) illustrates that the bits with large



(a) Real-valued (b) Binary (Sign) (c) Binary (Softsign) (d) Binary (BTL)

Figure 4. t-SNE visualizations of 16-bit binary descriptors with different binarization processes in the validation set.

value suffer the high penalty for the gradient $grad_{nk}^{(1)}$ and are greatly influenced by changes in other bits, which helps model converge (shown in Tab. 3).

Fig. 4 illustrates that BTL does help models learn binary descriptors directly and works well. We trained Resnet18 to obtain 16-bit descriptors on CIFAR10 and only changed the binarization method for comparison. Fig. 4a shows the distribution of real-valued descriptors as the benchmark. Through Figs. 4b to 4d, the distribution of binary descriptors with BTL is more discriminative.

3.2. Ultra-Wide Temperature-Scaled Cross Entropy Loss

We adopted the only loss function, contrastive loss, since BTL aligns the training goal with the final goal. The contrastive loss is based on the idea of InfoNCE [21] and is widely used in self-supervised learning. The previous visual representation methods based on self-supervised learning [5, 6, 12–14] are mainly designed for image classification. Their used NT-Xent loss is

$$\ell(i,j) = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{m \neq i} \exp(sim(z_i, z_m)/\tau)}. \quad (12)$$

Let $\mathrm{sim}(u,v) = u^T v / \|u\| \|v\|$ denote the cosine similarity. This formula is limited by the base e of exponentiation. As the temperature hyper-parameter τ decreases from 1 (τ is 1 or less in practice), the exponential function $exp(*)$ grows faster, making calculations more and more prone to exceed the numerical range supported by the data type. This limitation results in non-optimal performance of the binary descriptor in the patch-level tasks (as shown in Fig. 6) and perhaps more tasks. To support the various application of binary descriptors, we need a loss function that enables more diverse distribution of descriptors while ensuring the advantages of InfoNCE [21]. Therefore, our loss function is defined as

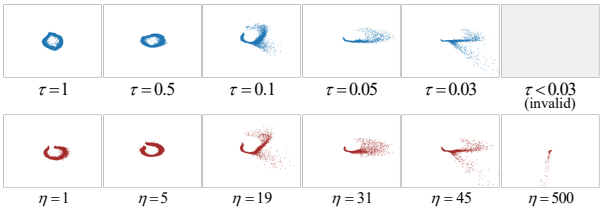


Figure 5. Comparing the distribution of descriptors with two loss functions in different hyper-parameters. *The top line*: the distributions of descriptors with the traditional NT-Xent loss in different hyper-parameter τ. *The bottom line*: the distributions of descriptors with our WUT-Xent loss in different hyper-parameter η.

$$\ell(i,j) = -\log \frac{[(a + sim(b_i, b_j))/div]^{\eta}}{\sum_{m \neq i} [(a + sim(b_i, b_m))/div]^{\eta}}, \quad (13)$$

where a and div are fixed, and their values are 1.1 and 2. The reason for this setup is that the results of exponentiations in Eq. (13) are from 0 to 1, while a and div are 1 and 2. To avoid the exponential result 0, we add a epsilon 0.1 to a. η is the only hyper-parameter to tune, which acts similar to the temperature hyper-parameter of SimCLR [6] but enables more diverse distribution. Fig. 5 supports this point: there are one-to-one correspondences between the first six sub-pictures in both lines, and each pair of sub-pictures shows a similar distribution. The minimum value τ can take while using NT-Xent loss is 0.03. Our loss function covers the adjustment range of the previous function when η is less than 50 and covers more than ten times the range, which means more diverse distribution.

The i^{th} view is the current anchor, and (i,j) is a positive pair of examples generated from the same image by random data augmentation. Their binary descriptors are encoded by the parameter-shared network as positive pairs, while the negative pairs are encoded from the views of different images. In a batch, a positive pair and $2N-2$ negative pairs compose all the input used in the loss function. The possible values of m are from 1 to $2N$ except i but including j. Overall, the total loss for our approach can be formulated as

$$Loss = \frac{1}{2N} \sum_{m=1}^{N} \ell(2m-1, 2m) + \ell(2m, 2m-1), \quad (14)$$

where $2m-1$ and $2m$ indicate that the current loss function's input is two views of the same image.

4. Experiments

In this section, we compared BTL with some general binarization processes and inserted BTL into the previous

Dataset	CIFAR10 [17]	Phototour [3]	Flower17 [19]
Type	32×32 images	64×64 patches	nature images
Label	10-class	pair-wise	17-class
Training	50,000	∼1,500,000	680
Test	10,000	300,000	340
Evaluation	mAP@1000	FPR@95	Accuracy

Table 1. Summary of the evaluated datasets in our experiments. ∼ indicates that the value is an approximation.

work to demonstrate the role of BTL. Also, we clarified the superiority of our loss function in patch-level tasks, tested the compatibility of DLBD by replacing different backbone networks and showed the role of the components in DLBD through an ablation study. The experiments on several public image-level and patch-level datasets shown in Tab. 1 demonstrate that our DLBD achieves state-of-the-art performance and generalization ablity.

4.1. Datasets and Evaluation Setups

In the experiments, we used three public datasets to evaluate the performance of our method on different visual tasks. CIFAR10 [17] is a single-label dataset. The previous unsupervised learning-based binary descriptors [8, 9, 15, 29] evaluated their retrieval performance on this image-level dataset, assessing the representation ability of high-dimensional features of the learned binary descriptors. UBC Phototour [3] is a standard patch-level dataset for evaluating learning-based patch descriptors on patch verification tasks. The patches of this dataset are extracted from images of three landmarks: Liberty, Notre Dame, and Yosemite. These methods [8, 9, 15, 29] utilized it to estimate the representation ability of low-dimension features of the learned binary descriptors. Flower17 [19] is a high-resolution single-label dataset used for flower recognition, a classic visual analysis task. This dataset is used to test the performance for fine-grained recognition. Tab. 1 shows the specifications of these datasets.

4.2. Implementation Details

In our framework, the two views of the same image as input are obtained through data augmentation. The data augmentation pipeline consists of random resized cropping, random color jittering, random rotation, and random horizontal flip. Given an input view, the feature maps can be extracted by any convolutional Neural Networks and then forwarded to the following BTL. In our experiments, we initialized models with the parameters pre-trained on ImageNet. By default, we trained our model with an initial learning rate of 0.001 and adjusted it by Cosine Annealing. The idea of choosing hyperparameters η is as follows: The smaller the difference between generated positives and ground-truth, the larger η should be. The positive samples were obtained through data augmentation rather than the ground-truth in contrastive learning. Usually, the performance curves (like Fig. 6b) will be convex. Based on the above analysis, we search for optimal parameters.

Binarization Process	mAP@1000 (%)↑		
	16-bit	32-bit	64-bit
sign	51.62	58.02	60.81
norm.^{0}+sign	53.26	58.72	61.38
norm.^{1}+sign	53.63	59.20	61.70
norm.^{0} (BLCD)+sign	54.37	58.33	61.55
norm.^{1} (BLCD)+sign	52.54	58.84	61.73
softsign	55.52	59.69	62.24
norm.^{0}+softsign	53.72	59.25	62.42
norm.^{1}+softsign	51.44	60.05	62.27
norm.^{0} (BLCD)+softsign	54.13	59.75	62.64
norm.^{1} (BLCD)+softsign	54.00	59.72	62.12
BTL	**58.58**	**61.51**	**62.88**

Table 2. The performance of binary descriptors generated by different binarization processes on cifar10. Norm.^{0} is batch-normalization and norm.^{1} is instance-normalization. Both are normalizations with variance 1 and mean 0. Norm. (BLCD) is one with variance $\sqrt{2/\pi}$ and mean 0, which is a good inductive bias (for sign) proposed in BLCD.

4.3. Experiments on BTL

We compared our BTL with the wide-used binarization process. This binarization process is widely used in DeepBit [15], BLCD [10], L2Net [23] and so on. The detailed comparison is shown in Tab. 2: BTL helps models reduce quantization loss more effectively than others. Moreover, we also demonstrated the generality of our BTL by inserting it into the previous model BLCD [10], showing improvements in all six sets of experiments. Note that BTL and other works focus on reducing the gap between the real-valued and binary descriptor. Therefore, the effect of BTL on binary descriptor is reduced. (BTL is a binarization used to decrease the performance loss while transforming real-valued descriptor methods into binary counterparts.)

Method	mAP@1000 (%)↑		
	16-bit	32-bit	64-bit
BTL w/o Eq. (6)	26.06	31.37	39.36
BTL w/ SG(Eq. (6))	39.39	45.61	49.48
BTL w/ Eq. (6)	58.58	61.51	62.88

Table 3. The ablation study of the proposed BTL on the CIFAR10 dataset. SG(Eq. (6)): BTL only follows the forward propagation in Eq. (6) (i.e. changing Eq. (5) to $grad_{nk}^{(2)} = grad_{nk}^{(1)}$).

Training	Testing	BLCD (Binary)↓	BLCD w/ BTL (Binary)↓
Yosemite	Liberty	12.12	**12.06**
Notredame		11.69	**10.65**
Yosemite	Notredame	7.94	**7.59**
Liberty		8.11	**7.31**
Notredame	Yosemite	11.18	**9.62**
Liberty		11.42	**10.13**
FPR@95 (mean)		10.41	**9.56**

Table 4. The FPR@95 of the binary descriptor learned by BLCD with our BTL. All the results come from the method we reproduce with the usual data augmentation.

Tab. 2 illustrates the superiority of BTL. As mentioned in Sec. 3.1, the superiority is due to the way that BTL allows the model to work well in backpropagation. The role of BTL in backpropagation is to adjust the sign and magnitude of gradient. The former has been demonstrated theoretically and the latter is further illustrated by Tab. 3.

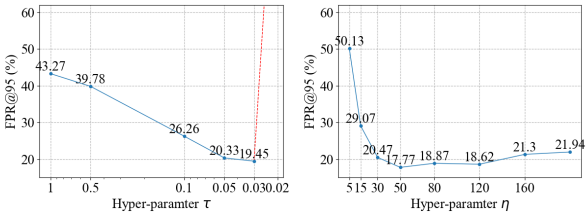
4.4. Experiments on UWT-Xent Loss

We conducted this experiment to illustrate that the idea of our loss function is practically required. As shown in Fig. 6a, the optimal distribution may not be realized for the limited hyper-parameter selection range. The performance of DLBD is still in an improvement trend when the hyper-parameter is adjusted from the maximum value to the minimum value of 0.03. There are many similar situations when solving tasks related to patch-level datasets. To cope with these situations, we presented the UWT-Xent loss, and the corresponding results are illustrated in Fig. 6b. The proposed UWT-Xent loss has a wider parameter selection range and thus can reach the optimal. It also shows a good property of hyper-parameter selection, given that the curves are usually convex. Since the UWT-Xent loss provides more distribution forms in representation space, we can get a better performance over DLBD.

4.5. Experiments on DLBD

Generality of DLBD. We show our DLBD with different backbones in Tab. 5, and give the quantization loss of DLBD with sign function as a baseline for comparison. DLBD with different backbones can work well. L2Net is a network that enables small-size input and only has about one-tenth of the parameters of ResNet18. Therefore, it is acceptable for its results.

Ablation study. As shown in Tab. 7, we modernize a SimCLR [6] towards DLBD, transforming the method for real values to binary without any extra design. This study clearly illustrates the role of our components, and BTL indeed reduces performance loss.



(a) NT-Xent loss (b) UWT-Xent loss

Figure 6. Comparison between the NT-Xent loss and the proposed UWT-Xent loss. The red dotted line in (a) represents that when the hyper-parameter is set to 0.02, the model yields random outputs and cannot converge during training.

Backbone	Bin.	mAP@1000(%)↑			mAP@1000 loss↓		
		16-bit	32-bit	64-bit	16-bit	32-bit	64-bit
ResNet18	w/o	61.58	63.46	63.35	-	-	-
	Sign	51.61	58.02	60.81	9.97	5.32	2.54
	BTL	58.58	61.51	62.88	**3.00**	**1.95**	**0.47**
VGG16	w/o	65.56	66.44	66.90	-	-	-
	Sign	57.98	62.99	65.17	7.58	3.45	1.73
	BTL	64.86	65.45	66.03	**0.70**	**0.99**	**0.87**
L2Net	w/o	48.63	48.99	48.92	-	-	-
	Sign	38.61	42.47	44.57	10.02	6.52	4.35
	BTL	44.87	49.21	51.24	**3.76**	**-0.22**	**-2.32**

Table 5. Illustration of the quantization loss (mAP@1000 loss) of DLBD with different backbones. mAP@1000 loss represents the performance penalty due to the binarization.

4.6. Comparative Experiments and Discussion

We mainly chose the recently proposed unsupervised learning for binary descriptors for comparison, such as BinGAN [29], GraphBit [9], DeepBit [15], DCBD-MQ [8], BLCD [10], and UDBD [26].

Results on Image Retrieval. The evaluated methods' results are listed in Table 8. In this experiment, we trained the network for 200 epochs. The best performance of the previous unsupervised binary descriptors is achieved by UDBD where mAPs@1000 are 32.24, 36.17, and 39.60 responding to 16-bit, 32-bit, and 64-bit vectors. Our DLBD achieves a great improvement in performance that mAPs@1000 are 58.58, 61.51, and 62.88, respectively. Note that we did not give the results of BLCD in this task, since BLCD is designed for patch-level tasks.

Results on Patch Matching. To compare with the previous methods, we conducted experiments on Phototour dataset, which is widely used to evaluate the performance of patch descriptors. Tab. 6 summarizes the FPR@95 of all the eval-

Method	Type	Dim	Yose.	Notre.	Yose.	Lib.	Notre.	Lib.	FPR95 (mean)↓
			Liberty↓		Notredame↓		Yosemite↓		
L2-Net [23]	SP	256-bit	7.83	5.25	3.52	3.07	6.92	8.49	5.84
BinGAN [29]	USP	256-bit	26.08	25.76	16.88	27.84	40.80	47.64	30.76
GraphBit [9]	USP	256-bit	24.72	21.18	17.78	15.25	49.94	49.64	29.75
DeepBit [15]	USP	256-bit	34.64	33.83	28.49	20.66	54.63	56.69	38.15
DCBD-MQ [8]	USP	256-bit	25.77	22.92	20.13	18.95	50.99	50.36	31.52
BLCD [10]	USP	256-bit	*11.90*	**10.07**	**5.26**	**4.90**	**10.03**	**9.02**	**8.53**
UDBD [26]	USP	256-bit	20.79	18.99	14.61	11.76	52.60	52.17	28.49
DLBD(η=90)	USP	256-bit	**11.25**	*11.44*	*7.01*	*8.55*	*20.58*	*20.23*	*13.18*

Table 6. The false positive rate at 95 percent recall rate (FPR95) of different descriptors on the Phototour dataset for patch matching. Italic font indicates the suboptimal performance of the unsupervised (USP) methods and BLCD is only applicable to patch-level images.

SimCLR	BTL	UWT.	mAP@1000/Accuracy (%)↑		
			16-bit	32-bit	64-bit
✓			60.8/66.6	62.3/69.1	62.3/69.2
✓		✓	61.6/66.6	63.5/70.0	63.4/70.8
✓	✓		57.3/64.8	61.0/67.9	62.1/68.3
✓	✓	✓	58.6/64.5	61.5/67.8	62.9/69.4

Table 7. The ablation study of the proposed DLBD on the CIFAR10 dataset. Note that descriptors without BTL are real-valued.

Method	mAP@1000 (%)↑		
	16-bit	32-bit	64-bit
BinGAN [29]	30.05	34.65	36.77
GraphBit [9]	27.79	33.45	37.97
DeepBit [15]	26.36	27.92	34.05
DCBD-MQ [8]	30.58	33.01	36.59
BLCD [10]	-	-	-
UDBD [26]	32.24	36.17	39.60
DLBD(ours, η=4)	**58.58**	**61.51**	**62.88**

Table 8. The mAP(%) at the top 1, 000 returned images of different descriptors on the CIFAR10 dataset for image retrieval.

uated methods, and our methods achieved 13.18 percent mean FPR@95, which is better than most previous unsupervised works. BLCD decreases to 8.53 percent, which achieves the best accuracy compared to other unsupervised methods on this dataset. However, BLCD is designed for patch-level applications, while our method can be used in both patch-level and image-level applications.

Results on Image Classification. In this experiment, we focused on recognizing the different subclasses of the same objective category, which requires the learned binary descriptors to have a strong representation ability for delicate features. Tab. 9 shows the performance of Deepbit and our approach for fine-grained recognition. We trained the multi-class SVM classifiers with the binary descriptors our approach learned and also introduced the widely used linear evaluation protocol [2,6,16,22,28], and the results in Tab. 9 shows the superiority of our method.

Method	Dim	Accuracy (%)↑
DeepBit (SVM)	512-bit	89.2
DLBD (SVM)	512-bit	93.6
DLBD (SVM)	64-bit	89.5
DLBD (Linear)	512-bit	94.0

Table 9. The Categorization Accuracy for different binary descriptors on the Flower17 dataset for image classification.

5. Conclusions

This paper presents a novel and model-agnostic Binary transformation Layer (BTL) to enable direct-learned binary descriptors. Based on the proposed BTL, we designed a self-supervised learning framework to directly learn binary descriptors, providing a new way to construct binary descriptors. In addition, we designed UWT-Xent loss that is suitable for binary descriptors and allows more diverse distribution of descriptors. Extensive experiments on the public datasets show that the proposed method outperforms the state-of-the-art methods in some applications.

Acknowledgements

This work was supported in part by the National Key Research and Development Project under Grant 2019YFE0110800, in part by the National Natural Science Foundation of China under Grant 62172067 and 61976031, in part by the National Major Scientific Research Instrument Development Project of China under Grant 62027827.

References

[1] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE conference on computer vision and pattern recognition*, pages 510–517. Ieee, 2012. 1, 2

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019. 8

[3] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):43–57, 2010. 6

[4] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007. 1

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 2, 5

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2, 5, 7, 8

[7] Ling-Yu Duan, Jie Lin, Zhe Wang, Tiejun Huang, and Wen Gao. Weighted component hashing of binary aggregated descriptors for fast visual search. *IEEE Transactions on multimedia*, 17(6):828–842, 2015. 1

[8] Y. Duan, J. Lu, Z. Wang, J. Feng, and J. Zhou. Learning deep binary descriptor with multi-quantization. *IEEE Trans Pattern Anal Mach Intell*, 41(8):1924–1938, 2019. 1, 2, 6, 7, 8

[9] Yueqi Duan, Ziwei Wang, Jiwen Lu, Xudong Lin, and Jie Zhou. Graphbit: Bitwise interaction mining via deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8270–8279, 2018. 1, 6, 7, 8

[10] Bin Fan, Hongmin Liu, Hui Zeng, Jiyong Zhang, Xin Liu, and Junwei Han. Deep unsupervised binary descriptor learning through locality consistency and self distinctiveness. *IEEE Transactions on Multimedia*, pages 1–1, 2020. 1, 2, 3, 6, 7, 8

[11] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465. IEEE, 2005. 1

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, and Mohammad Gheshlaghi Azar. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 2, 5

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 2, 5

[14] Peng Hu, Xi Peng, Hongyuan Zhu, Liangli Zhen, and Jie Lin. Learning cross-modal retrieval with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5403–5413, June 2021. 2, 5

[15] Lin Kevin, Lu Jiwen, Chen Chu-Song, Zhou Jie, and Sun Ming-Ting. Unsupervised deep learning of compact binary descriptors. *IEEE Trans Pattern Anal Mach Intell*, 41(6):1501–1514, 2019. 1, 2, 3, 6, 7, 8

[16] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929, 2019. 8

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[18] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011. 1, 2

[19] M-E Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1447–1454. IEEE, 2006. 6

[20] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002. 1, 2

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 5

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 8

[23] Yurun Tian, Bin Fan, and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 661–669, 2017. 1, 2, 6, 8

[24] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2881, 2013. 1, 2

[25] Ziwei Wang, Quan Zheng, Jiwen Lu, and Jie Zhou. Deep hashing with active pairwise supervision. In *European Conference on Computer Vision*, pages 522–538. Springer, 2020. 1

[26] Gengshen Wu, Zijia Lin, Guiguang Ding, Qiang Ni, and Jungong Han. On aggregation of unsupervised deep binary descriptor with weak bits. *IEEE Transactions on Image Processing*, 29:9266–9278, 2020. 1, 7, 8

[27] Jianming Ye, Shiliang Zhang, Tiejun Huang, and Yong Rui. Cdbin: Compact discriminative binary descriptor learned with efficient neural network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(3):862–874, 2019. 1, 2

[28] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 8

[29] Maciej Zieba, Piotr Semberecki, Tarek El-Gaaly, and Tomasz Trzcinski. Bingan: Learning compact binary descriptors with a regularized gan. *arXiv preprint arXiv:1806.06778*, 2018. 1, 3, 6, 7, 8