TransHash: Transformer-based Hamming Hashing for Efficient Image Retrieval

Yongbiao Chen*
chenyongbiao0319@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Sheng Zhang
zhangshe@usc.edu
University of Southern California
Los Angeles, United States

Fangxin Liu
liufangxin@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Zhihang Chang*
changz@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Mang Ye
mangye16@gmail.com
Wuhan University
Wuhan, China

Zhengwei Qi
qizhwei@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Deep hashing has gained growing popularity in approximate nearest neighbour search for large-scale image retrieval. Until now, the deep hashing for the image retrieval community has been dominated by convolutional neural network architectures, e.g. ResNet [21]. In this paper, inspired by the recent advancements of vision transformers, we present TransHash, a pure transformer-based framework for deep hashing learning. Concretely, our framework is composed of two major modules: (1) Based on Vision Transformer (ViT), we design a siamese vision transformer backbone for image feature extraction. To learn fine-grained features, we innovate a dual-stream feature learning on top of the transformer to learn discriminative global and local features. (2) Besides, we adopt a Bayesian learning scheme with a dynamically constructed similarity matrix to learn compact binary hash codes. The entire framework is jointly trained in an end-to-end manner. To the best of our knowledge, this is the first work to tackle deep hashing learning problems without convolutional neural networks (CNNs). We perform comprehensive experiments on three widely-studied datasets: CIFAR-10, NUSWIDE and IMAGENET. The experiments have evidenced our superiority against the existing state-of-the-art deep hashing methods. Specifically, we achieve 8.2%, 2.6%, 12.7% performance gains in terms of average mAP for different hash bit lengths on three public datasets, respectively.

CCS CONCEPTS
• Computer systems organization → Embedded systems; Redundancy; Robotics; • Networks → Network reliability.

KEYWORDS

hashing hashing, deep learning, image retrieval, vision transformer

1 INTRODUCTION

The past decade has been characterized by the explosive amount of high-dimensional data generated by countless end-users or organizations, resulting in a surge of research attention on accurate and efficient information retrieval methods. Among them, large-scale image retrieval has attained growing traction for its pervasive uses in various scenarios, e.g. recommendation systems, search engines, remote sensing systems. Among all the methods proposed for this challenging task [16, 17, 27, 38], hashing-based methods have achieved pronounced successes. It aims to learn a hash function mapping the images in the high-dimensional pixel space into low-dimensional hamming space while preserving their visual similarity in the original pixel space. Scores of works have been introduced. Based on the way they extract features, existing hashing-based works can be divided into two categories, namely, shallow methods and deep learning-based methods. Shallow methods [8, 26, 47] learn their hash functions via the hand-crafted visual descriptors (e.g. GIST [39]). Nonetheless, the handcrafted features do not guarantee accurate preservation of semantic similarities of raw image pairs, resulting in degraded performances in the subsequent hash function learning process. Deep learning-based [14, 48] methods generally achieve significant performance improvements when compared to their shallow counterparts. The common learning paradigm involves two phases. The first phase aims to learn discriminative feature representations with deep convolutional neural networks (CNNs), e.g. AlexNet. The second phase involves designing diversified non-linear functions to squash the continuous features into binary Hamming codes and devising various [4, 5, 15, 20, 34] losses to preserve the similarity in the raw pixel space.

Recently, transformers [45] have demonstrated great successes in natural language processing [3, 11]. With the advent of the Vision Transformer, a variant of transformer tailored for computer vision tasks, transformers have trumped numerous CNN-based methods in various computer vision tasks (e.g. image classification [13], object...
At last, we add a hash layer projecting the feature into $B$-bit hash vectors. The Bayesian learning module is employed to preserve the similarity in the hashing space for each pair.

Re-identification [23], and etc). As is shown in Fig. 1, Vision transformer works by first reshaping the input images into a sequence of 2D patches. In the later stage, the 2D patches are transformed into $D$ dimensional vectors with a trainable linear projection matrix. Then, a sequence of 1D vectors is fed into the standard transformer architecture to learn a usable feature representation. Inspired by the pronounced performances of ViT in other vision tasks, we ponder the possibility of innovating novel deep hashing methods with pure transformers.

In this paper, we build up a novel transformer-based hashing method, dubbed Transhash, which is the very first deep hashing method without adopting a convolutional neural network (CNN) as the backbone architecture. Specifically, targeting pairwise deep hashing learning, we design a Siamese Transformer backbone, which is essentially two identical transformers sharing the same weight parameters[2]. On top of this innovation, inspired by [23], we design a dual-stream feature learning module by changing the last layer of two Siamese transformers to two parallel branches. Concretely, for the first branch, we learn a global feature representation while reducing bit redundancy.

To sum up, we make the following contributions:

1. We design a Siamese Transformer backbone based on ViT which is two identical vision transformers sharing the same weight parameters.
2. We innovate a novel two-stream feature learning module by changing the last layer of the transformer into two independent parallel branches. In this fashion, we could learn global and local features at the same time. Meanwhile, as stated before, it could also promote the independence of the learned final hash code vector while reducing bit redundancy.
3. By further adopting the similarity-preserving Bayesian learning module with a quantization constraint, we build up a novel deep hashing framework for large-scale image retrieval with pure transformer. To the best of our knowledge, this is the very first work for deep learning-based hashing without adopting a convolutional neural network as the backbone.
4. We conduct comprehensive experiments on three widely-studied datasets - CIFAR-10, NUSWIDE and IMAGENET. The results show that we outperform all the state-of-the-art method across three datasets by large margins.

2 RELATED WORKS

2.1 CNNs in Computer Vision

Convolutional neural network was first introduced in [32] to recognize hand-write numbers. It proposes convolutional kernels to capture the visual context and achieves notable performances. Nonetheless, it was not until the innovation of AlexNet [29] that the CNN starts to become the workhorse of almost all the mainstream computer vision tasks, e.g. Instance Segmentation [1, 40], Image Inpainting [52, 53], Deep hashing [5, 6], Person Re-identification [9, 25, 50, 51] and etc. To further boost the capability of CNNs, a series of deeper and more effective convolutional neural networks have been proposed, e.g. VGG [42], GoogleNet [43], ResNet [22], EfficientNet [44] and etc. While CNNs are still dominant across various computer vision tasks, the recent shift in attention to transformer-based architectures has opened up possibilities to adopt transformers as potent alternatives to convolutional neural networks. Our work is...
among the first endeavours to replace CNNs with pure transformer-based architectures in traditional computer vision tasks.

2.2 Transformer in Vision

The Transformer is first proposed in [45] for sequential data targeting at usage in the field of natural language processing (NLP). Since then, many studies have investigated the effectiveness of Transformer in computer vision tasks by feeding to Transformer the sequence of feature maps extracted by CNNs [7, 18, 49]. In 2020, Google proposed Vision Transformer (ViT) [13], which applies a pure transformer directly to a sequence of image patches for image classification. Variants of ViT have achieved remarkable successes. For instance, [37] proposes a hierarchical vision transformer using shifted windows. [46] proposes a pyramid vision transformer tailored for dense prediction. Further, [23] proposes the first work of designing a pure-transform based architecture for person re-identification. By utilizing the side information and innovating a novel jigsaw branch, it achieves state-of-the-art across multiple object re-identification datasets. Vision transformer is still in its nascent stages. Mounting research attention is being directed to investigate its potential in diversified computer vision tasks.

2.3 Hashing for Image Retrieval

Deep hashing for large-scale image retrieval has been drawing growing research attention in recent years [17, 19, 24, 26, 27, 47]. According to the way they extract the features, we could categorize the existing hashing methods into two groups: shallow hashing methods and deep learning based-hash methods.

Typical shallow methods are reliant upon the handcraft features to learn a hashing function mapping visual images into binary hash codes. A canonical example is LSH (Locality Sensitive Hash) [26], which seeks to find a locality-sensitive hash family where the probability of hash collisions for similar objects is much higher than those dissimilar ones. Later, [8] further proposed another variant of LSH (dubbed SIMHASH) for cosine similarities in Euclidean space. Though these handcrafted feature-based shallow methods...
achieved success to some extent, when applied to real data where dramatic appearance variation exists, they generally fail to capture the discriminative semantic information, leading to compromised performances. In light of this dilemma, a wealth of deep learning-based hash methods have been proposed [6, 15, 33, 34, 55], for the first time, proposes to learn the features and hash codes in an end-to-end manner. [55] offers a Bayesian learning framework adopting pairwise loss for similarity preserving. [5] further suggests substituting the previous probability generation function for neural network output logits with a Cauchy distribution to penalize similar image pairs with hamming distances larger than a threshold. [54] innovates a new similarity matrix targeting multi-label image retrieval. [15] further introduces a deep polarized loss for the hamming loss retrieval. [15] further introduces a deep polarized loss for the hamming loss retrieval. [15] further introduces a deep polarized loss for the hamming loss retrieval. [15] further introduces a deep polarized loss for the hamming loss retrieval. [15] further introduces a deep polarized loss for the hamming loss retrieval.

### 3 Proposed Method

#### Problem Formulation
Suppose we have a training set $T = \{I_i\}_{i=1}^{NT}$ containing $NT$ training images and the corresponding label set $Y = \{y_i\}_{i=1}^{NT}$. For all the pairs of images in the training set, we can construct a similarity matrix $S$ where $s_{ij} = 1$ if $I_i$ and $I_j$ are from the same class and $s_{ij} = 0$ otherwise. The goal of deep hash for image retrieval is to learn a non-linear hash function $H : I \mapsto \{0, 1\}^B$ which encodes each input image $I_i$ into a binary hash vector $h_i$ with B bits while preserving the similarity information conveyed in $S$. That is to say, the Hamming distance between $h_i$ and $h_j$ should be small if $s_{ij} = 1$ and large otherwise.

#### 3.1 Siamese Vision Transformer Architecture

An overview of our architecture is illustrated in Fig. 2. For an image pair $(I_i, I_j)$ with size $H \times W \times 3$, we cut them into identical small patches of patch size $P \times P \times 3$. In doing so, we obtain $N$ patches in total, where $N = H \times W / P^2$. Note that $N$ is also the effective input sequence length for the transformer.

**Patch embeddings.** For each image patch of $P \times P \times 3$, we flatten it into a vector of size $P^2 \times 3$. Subsequently, similar to ViT, we embed every vector into $D$ dimensions with a trainable linear projection (fully connected layer), resulting in a sequence $\{x_{ik}^p\} \in \mathbb{R}^D, k \in [1, N]$. We further prepend a learnable embedding $x_{class}$ to $x$, whose state at the end of the output layer serves as the image representation. In this way, we obtain the final embedding $X_p \in \mathbb{R}^{(N+1) \times D}$.

**Position embeddings.** Positional embedding is adopted to encode the position information of the patch embedding, which is important for the transformer to learn the spatial information of each patch inside the original image. We follow the standard procedure in ViT by adding trainable 1D position embedding for every vector in the sequence. Thus, the input for the transformer encoder is stated as follows:

$$z_0 = X_p + E_{pos} = [x_{class}, x_{p}^1, ..., x_{p}^N] + E_{pos}$$

**Self-attention encoder.** The transformer encoder consists of $L - 1$ blocks, each block containing a multi-headed self-attention layer (MSA) and MLP layer. A layer norm (LN) is applied before each layer while residual connections are applied after each layer, as shown in Fig. 2. The computation of a block $\mathcal{F}_{block}$ could be formulated as:

$$z_l = \mathcal{F}_{msa}(\mathcal{F}_{ln}(z_{l-1})) + z_{l-1}$$
$$z_l = \mathcal{F}_{mlp}(\mathcal{F}_{ln}(z_i)) + z_l$$

where $l = 1,...(L - 1)$

**Dual-stream feature learning.** After the before-mentioned self-attention encoder, we get the hidden features which are denoted as $Z_{l-1} = [z_{l-1}, z_{l-1}, z_{l-1}, ..., z_{l-1}]$. Note that, as stated before, $z_{l-1}$ is the hidden feature for the prepended learnable embedding $x_{class}$. Inspired by [23], we design two parallel branches, the global branch $\mathcal{F}_{block}^g$ and the local branch $\mathcal{F}_{block}^l$. For the global branch, it serves as a standard transformer block encoding $Z_{l-1}$ into $Z_l = [f_{g}^1, z_{l-1}^1, z_{l-1}^2, ..., z_{l-1}^N]$, where $f_{g}$ is regarded as the global feature representation. For the local branch, we split $Z_{l-1}$ into $K$ groups and prepend the shared token $z_{l-1}^0$ before each group. In this way, $K$ feature groups are derived which are denoted as $\{z_{l-1}^0, z_{l-1}^1, ..., z_{l-1}^{N/K}\}$ and $\{z_{l-1}^0, z_{l-1}^1, ..., z_{l-1}^{N/K+1}\}$. Then, we feed $K$ features groups into $\mathcal{F}_{block}^l$ to learn $K$ local features $\{f_{l}^1, f_{l}^2, ..., f_{l}^K\}$.

**Hash layer.** In an effort to learn compact hash codes, we further design several hash layers projecting every feature vector into different bit sized hash vectors. Concretely, suppose the hash bit length in the retrieval stage is $B$ for each image, then, for the global feature vector of embedding size $M$, we obtain a $B/2$ bit global hash vector through

$$h_g = \mathcal{F}_h^g(f_g) = f_g W^T + b$$

where $W$ is a weight parameter matrix of size $(B/2, M)$ and $b$ is the bias parameter of size $(B/2, )$. In a similar fashion, for each local feature $f_l \in \{f{l}^1, ..., f{l}^K\}$, we design a specific fully connected layer with $B/(2 \times K)$ output logits, resulting in $K$ hash vectors $\{h{l}^1, ..., h{l}^K\}$.

In this way, for an image pair $(I_i, I_j)$, the Siamese model outputs two sets of hash vectors: $\{h_g^1, h_g^2, ..., h_g^K\}$ and $\{h{l}^1, h{l}^2, ..., h{l}^K\}$, respectively.

#### 3.2 Similarity-preserving Bayesian Learning

In this paper, we propose to adopt a Bayesian learning framework for similarity-preserving deep hashing learning. Given training images $(I_i, I_j, s_{ij}) : s_{ij} \in S$, where $s_{ij} = 1$ if $I_i$ and $I_j$ are from the same class and $0$ otherwise, we can formulate the log posterior Maximum a Posteriori (MAP) estimation of the hash codes $H = \{h_1, h_2, ..., h_p\}$ for $P$ training points as:

$$\log P(H \mid S) \propto \log P(S \mid H)P(H)$$

$$= \sum_{s_{ij} \in S} w_{ij} \log P(s_{ij} \mid h_i, h_j) + \sum_{i=1}^{NT} \log P(h_i)$$

where $P(S \mid H)$ is the weighted likelihood function and $w_{ij}$ is the corresponding weight for each image pair $(I_i, I_j)$. Since the similarity matrix $S$ could be very sparse in real retrieval scenarios [6], it could lead to the data imbalance problem, resulting in sub-optimal
We are increasing the value of where we adopt a surrogate \( D \) where \( S \) denotes the scale parameter of the Cauchy distribution. We propose to adopt \( \sigma \) is a probability function which takes as input a distance of a hash code pair and generate the probability of them from the same class. Note that, since directly optimizing the discrete binary hash code is super challenging, in the training stage, we apply continuous relaxation to the binary constraints \( h_i \in \{-1, 1\}^\beta \) similar to [5, 6, 55]. Thus, we adopt a surrogate \( D_S \) for \( D_H \) in the continuous space which is formulated as:

\[
D_S (h_i, h_j) = \frac{K}{2} \left( 1 - \cos (h_i, h_j) \right)
\]

For the probability function \( \sigma \), the most commonly used is the sigmoid function. Nevertheless, as stated in [5], the probability of sigmoid when the input Hamming distance is much larger than 2 stays high and only starts to decrease when it approaches \( h/2 \). This property makes it hard for the deep hashing method to pull the distance of similar pairs close to a sufficient amount. In light of this dilemma, we propose to adopt Cauchy distribution function:

\[
\sigma (D_S (h_i, h_j)) = \frac{1}{1 + D_S (h_i, h_j)}
\]

where \( \gamma \) denotes the scale parameter of the Cauchy distribution. The Cauchy distribution has a desirable property. The probability of Cauchy declines very fast even when the Hamming distance is small, enabling the hashing method to pull the similar images into a small Hamming radius. By taking Eq. 8, Eq. 7, Eq. 6 into the MAP estimation in Eq. 4, we could derive the optimization objective of similarity-preserving loss as:

\[
L_S = \sum_{sij \in S} L_{ce} (h_i, h_j) = \sum_{sij \in S} \omega_{ij} \left( s_{ij} \log \frac{D_S (h_i, h_j)}{\gamma} + \log \left( 1 + \frac{1}{D_S (h_i, h_j)} \right) \right)
\]

From Eq. 6 and Eq. 9, we can observe that \( L_S \) takes a similar form as logistic regression. By optimizing \( L_S \), for a similar pair \( (i, j) \), we are increasing the value of \( P (1 | h_i, h_j) \), resulting in decreased value of \( D_S (h_i, h_j) \) since \( \sigma \) is a monotonically decreasing Cauchy function.

The quantization constraint to bridge the gap between continuous features and their binary counterparts \( (L_Q) \) can be derived from the proposed prior \( P (h) = \frac{1}{1 + D_S (h, 1)} \) where \( \gamma \) is the same scale parameter as Eq. 8 and 1 is a vector of ones. Since we are maximizing \( P(H) \) in Eq. 4, the quantization loss \( L_Q \) is stated as:

\[
L_Q = \sum_{i=1}^{N_T} Q(h_i) = \sum_{i=1}^{N_T} \log \left( 1 + \frac{D_S (h_i, 1)}{\gamma} \right)
\]

where \( 1 \) is a vector of ones. By minimizing the quantization loss \( Q \) in the training stage, each dimension of the hash vector \( h \) is pushed to approximate 1.

### 3.3 End to End training

In this section, we will derive the overall optimization objective of our proposed Transhash method based on Sec. 3.1 and Sec. 3.2. Given training images in pairs such as \( (I_i, I_j) \), we obtain a pair of continuous hash vector sets \( \{h_y I_i, h_y I_j\}, \ldots, \{h_y K, h_y K\} \} \) and \( \{h_y I_i, h_y I_j\}, \ldots, \{h_y K, h_y K\} \} \) through the siamese vision transformer. Subsequently, for the local features, we obtain the Bayesian loss and quantization loss as:

\[
L_{local} = \sum_{k} \sum_{i} L_{ce} (h_y k I_i, h_y k I_j)
\]

where \( N \) is the total number of training images, \( S \) represents the similarity matrix, and \( K \) denotes the number of local features for each image. In a similar fashion, we could derive the losses for the global features. The overall learning objective for Transhash is formulated as:

\[
\min_{\theta} L_{global} + \lambda L_{local} + \mu (L_{Q, global} + L_{Q, local})
\]

where \( \theta \) denotes the set of parameters of the framework, and \( \lambda \) is the hyper-parameter for controlling the importance of the Cauchy quantization loss.
Table 1: Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on Three Datasets

| Datasets         | CIFAR-10@54000 | NUSWIDE@5000 | IMAGENET@1000 |
|------------------|----------------|--------------|---------------|
| Methods          | 16 bits 32 bits 48 bits 64 bits | 16 bits 32 bits 48 bits 64 bits | 16 bits 32 bits 48 bits 64 bits |
| SH [47] (NeurIPS) | -               | -            | -             |
| ITQ [19] (TPAMI)  | -               | -            | -             |
| KSH [36] (CVPR)   | -               | -            | -             |
| BRE [30] (NeurIPS)| -               | -            | -             |
| DSH [35] (CVPR)   | 0.6145 0.6815 0.6828 0.6910 | 0.6338 0.6507 0.6664 0.6856 | 0.4025 0.4914 0.5254 0.5845 |
| DHN [55] (AAAI)   | 0.6544 0.6711 0.6921 0.6737 | 0.6471 0.6725 0.6981 0.7027 | 0.4139 0.4365 0.4680 0.5018 |
| HashNet [6] (ICCV)| 0.5105 0.6278 0.6631 0.6826 | 0.6821 0.6953 0.7193 0.7341 | 0.3287 0.5789 0.6365 0.6656 |
| DCH [5] (CVPR)    | 0.6680 0.6936 0.6807 0.6775 | 0.7036 0.7178 0.7106 0.7056 | 0.5868 0.5862 0.5639 0.5540 |
| IDHN [54] (TMM)   | 0.5419 0.5695 0.5895 0.5972 | 0.6999 0.7149 0.7225 0.7256 | 0.2583 0.3339 0.3708 0.4037 |
| DPN [15] (IJCAI)  | 0.825 0.838 0.830 0.829  | -             | -             |
| TransHash         | 0.9075 0.9108 0.9141 0.9166 | 0.7263 0.7393 0.7532 0.7488 | 0.7852 0.8733 0.8932 0.8921 |

Figure 3: The experimental results of TransHash and other competing methods on three datasets

3.5 Implementation Details
All the images are first resized to 256 x 256. For the training images, we adopt standard image augmentation techniques including random horizontal flipping and random cropping with cropping size 224. For testing images, we only apply the center cropping with cropping size 224. The batch size is set to 64. SGD optimizer is adopted with a weight decay of 1e-4. The learning rate is initialized to 3e-2 with cosine learning rate decay. The number of warmup steps for the scheduler is set to 500. The patch size is set to (32, 32) for the Siamese transformer model, the hidden size to 1024. The number of heads for the multi-head attention is set to 16, and the model consists of 24 blocks in total.

4 EXPERIMENTATION

4.1 Datasets and Evaluation Protocols
Datasets. We conduct experiments on three widely-studied image retrieval datasets: CIFAR-10, NUSWIDE, and IMAGENET. CIFAR-10 [28] is a dataset with 60,000 images from 10 classes. We
follow the standard protocol in [5, 55]. Specifically, we randomly select 500 images for each class as the training set, resulting in 5,000 training points. Then, we randomly select 100 images per class as the query set, the rest denoted as the database.

**NUSWIDE** [10] is a widely-studied public web image dataset consisting of 269,648 images in total. Each image is annotated with some of the 81 ground-truth categories (concepts). For fair comparisons, we follow similar experimental protocols [6, 55] by randomly sampling 5,000 as the query set, the rest as the database. Subsequently, we randomly sample 10,000 images from the database as the training set.

**IMAGENET** is a subset of the dataset for Large Scale Visual Recognition Challenge (ISVRC 2015) [41]. Specifically, we follow the same protocol as [15][6] by randomly sampling 100 classes and use all the images of these classes in the validation set as the query set. All the images of these classes in the training set are denoted as the database, while 100 images per category are sampled as the training set.

**Evaluation Protocols.** We adopt Mean Average Precision (mAP), Precision and Recall as the testing metrics. Concretely, we follow a similar fashion as [5, 6]. The mAP is calculated with the top 54,000 returned images for **CIFAR-10**, 5,000 for **NUSWIDE** and 1,000 for **IMAGENET**

### 4.2 Comparison with State-of-the-Arts

In this section, we compare the results of our proposed **TransHash** and the state-of-the-art deep hashing methods. Specifically, the competing methods could be divided into two categories: shallow hashing methods and deep hashing methods. For the shallow hashing methods, we include the most frequently compared methods **SH** [47], **ITQ** [19], **KSH** [36], and **BRE** [30] for detailed comparisons. For the deep learning-based hashing methods, we further include **DSH** [35] which is among the very first works targeting at tackling the hashing problem for image retrieval with deep convolutional neural networks. In addition, we incorporate other recent deep hashing methods including **DHN**[55], **HashNet** [6], **IDHN** [54] and **DPN** [15].

Note that, for all the non-deep methods and **DPN**, we directly quote the results from [6] and [15]. For the rest of the competing methods, we conduct experiments with the open-sourced codes from the original papers. For fair comparisons, we conform to original protocols for the hyper-parameters and the pre-processing techniques. For example, all the images are resized to $224 \times 224$.

The Mean Average Precision (mAP) results are demonstrated in Tab. 1. It is rather evident that our proposed **TransHash** is a clear winner compared with the shallow hashing methods across three datasets. Specifically, we achieve absolute performance boosts of 19.93%, 39.69% in terms of average mAP for **NUSWIDE** and **IMAGENET**, respectively. The unsatisfied performances of these non-deep hashing methods could be in part attributed to the fact that these methods could not assist in the discriminative feature learning process, resulting in the generation of sub-optimal hashing codes. Clearly, deep hashing methods exhibit significantly better performances across all the datasets for different hash bit lengths. Still, our method outperforms all the competing methods by large margins. Specifically, on **CIFAR-10**, we achieve a mAP of 91.66% in terms of 64 hash bits, surpassing the state-of-the-art result by 8.8%. The performance improvement is even more pronounced in **IMAGENET**. The average mAP for **TransHash** is 86.10%, exceeding **DPN** by 12.7%. The reasons for the notable performance gains are twofold. First, the siamese architecture and the dual-stream feature learning design could assist in learning more discriminative features. The second reason is that the ratio between the number of similar pairs and dissimilar pairs in **IMAGENET** is much larger than **CIFAR-10**, which is also known as the data imbalance problem [6], deteriorating the performance of methods trained on pairwise data [35, 54]. **TransHash** tackles this problem by dynamically assigning a weight for each pair as is carried out in [6]. On **NUSWIDE**, our method also consistently exceeds the competing methods across different hash bit lengths. The performance gains are not as sizable as on **CIFAR-10** and **NUSWIDE** mainly because **TransHash** is not tailored for multi-label image retrieval where each image comprises multiple labels.

We further plot the Precision-Recall curves(PR) in terms of 16 and 64 hash bits and Precision curves with respect to different numbers of top returned images. As depicted in Fig. 3, the performance of **TransHash**, colored with red, consistently levitates above all the competing methods by large margins for the PR curves. In terms of precision w.r.t numbers of returned images, as shown in the top right pictures in Fig. 3, **TransHash** achieves significantly better results against all the methods. The results on **NUSWIDE** are on the middle of Fig. 3. **TransHash** achieves slightly better results for PR@16 bits and PR@64 bits. For the precision w.r.t number of returned images, our method obtains a precision of 76.77% for 100 returned images, surpassing **IDHN** by 2.7%. Pronounced performance gains could also be spotted for **IMAGENET**. Specifically, for PR curve with 16 bits, **DCH** obtains the second place while **HashNet** tops **DCH** for 48 bits. It is easy to spot that **TransHash** still exceeds both methods in two testing scenarios with considerable margins. For the precision curve, we achieve performances of 90.35%, 89.38% w.r.t 100 and 1000 returned images, exceeding **HashNet** by 24.73% and 28.18%, respectively. The superior results could sufficiently demonstrate the effectiveness of our pure-transformer-based hashing method.

### 4.3 Ablation Studies

To further analyze the overall design of our proposed method, we conduct a detailed ablation study to demonstrate the effectiveness of each component. Specifically, we investigate three variants of **TransHash**:

1. **TransHash w/o P**, a variant without adopting the dual-stream feature learning.
2. **TransHash w/o Q**, a variant without the Cauchy quantization loss.
3. **TransHash w/o C**, a variant adopting the sigmoid function as the probability function $\sigma$, following the protocols in [55].

As shown in Tab. 2 and Fig. 4, when the Cauchy quantization loss is removed (**TransHash w/o Q**), we experience notable performance declines in **NUSWIDE** and **IMAGENET**, from 74.88% to 69.15% and 89.21% to 87.58 % for 64 hash bits, respectively. When the model is deprived of Cauchy distribution (**TransHash w/o C**), which is similar to [55], we can see that the performance decreases
Table 2: Mean Average Precision (MAP) of Different Variants of TransHash on Three Datasets

| Datasets          | CIFAR-10@50000 | NUSWIDE@50000 | IMAGENET@1000 |
|-------------------|----------------|--------------|---------------|
|                   | 16 bits | 32 bits | 48 bits | 64 bits | 16 bits | 32 bits | 48 bits | 64 bits | 16 bits | 32 bits | 48 bits | 64 bits |
| TransHash         | 0.9075  | 0.9108  | 0.9141  | 0.9166  | 0.7263  | 0.7393  | 0.7532  | 0.7488  | 0.7852  | 0.8733  | 0.8932  | 0.8921  |
| TransHash w/o C   | 0.8406  | 0.8384  | 0.8958  | 0.9062  | 0.7004  | 0.7265  | 0.7336  | 0.7310  | 0.7172  | 0.7808  | 0.8064  | 0.8244  |
| TransHash w/o P   | 0.9029  | 0.9053  | 0.9028  | 0.9014  | 0.7190  | 0.7147  | 0.7339  | 0.7167  | 0.7549  | 0.8485  | 0.8635  | 0.8655  |
| TransHash w/o Q   | 0.8927  | 0.9023  | 0.9048  | 0.9078  | 0.6540  | 0.6821  | 0.6689  | 0.6915  | 0.7451  | 0.8588  | 0.8689  | 0.8758  |

Figure 4: Experimental results of different variants of TransHash on three datasets

Table 3: Analysis of the effects of K on CIFAR-10. Note that – denotes when K equals a certain number, the model fails to converge as illustrated in the empirical analysis.

| Groups (K) | 2    | 3    | 4    | 5    | 6    |
|------------|------|------|------|------|------|
| 16 bits    | 0.9075 | -    | -    | -    | -    |
| 32 bits    | 0.9108 | 0.9013 | -    | -    | -    |
| 48 bits    | 0.9141 | 0.9017 | 0.9187 | 0.9107 | 0.9143 |
| 64 bits    | 0.9166 | 0.9103 | 0.9037 | 0.9062 | 0.8994 |

Empirical analysis of K. As depicted in Tab. 3, generally, the performance is not very sensitive to K. Also, we observe that when the local feature vector is responsible for generating less than 4 bits, the model will fail to converge. In light of the above observations, we empirically set the K to 2 across four different hash bit lengths.

5 CONCLUSION

In this paper, we have proposed a novel pure transformer-based deep hashing framework (TransHash) to tackle the challenging large-scale image retrieval problem. Specifically, we innovate a novel Siamese transformer architecture for extracting robust image features with pairwise similarity learning. On top of that, in an attempt to learn more fine-grained features, we propose to add a dual-stream feature learning module to learn global and local features simultaneously. A well-specified Bayesian learning framework is adopted on top of all the pairwise features for similarity-preserving learning. The overall framework is optimized in an end-to-end fashion. We conduct extensive experiments and demonstrate that TransHash yields notable performance gains compared to the state-of-the-art deep hashing methods on CIFAR-10, NUSWIDE and IMAGENET datasets.
[51] Mang Ye, Zheng Wang, Xiangyuan Lan, and Pong C Yuen. 2018. Visible thermal person re-identification via dual-constrained top-ranking. In IJCAI. Vol. 1. 2.

[52] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. 2017. Semantic image inpainting with deep generative models. In Proceedings of the IEEE conference on computer vision and pattern recognition. 5485–5493.

[53] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 4471–4480.

[54] Zheng Zhang, Qin Zou, Yuewei Lin, Long Chen, and Song Wang. 2019. Improved deep hashing with soft pairwise similarity for multi-label image retrieval. IEEE Transactions on Multimedia 22, 2 (2019), 540–553.

[55] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep hashing network for efficient similarity retrieval. In Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30.