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

Zhigang Chang
changzig@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 neighbor 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`[22]. 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 **M**ulti-**G**ranular **V**ision **T**ansformer backbone (**MGVT**) for image feature extraction. To learn fine-grained features, we innovate a dual-stream multi-granular 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

• **Computing methodologies** → **Computer vision problems**; **Image representations**; • **Information systems** → **Expert search**.

KEYWORDS

deep hashing, deep learning, image retrieval, vision transformer

ICMR '22, June 27–30, 2022, Newark, NJ, USA

ACM ISBN 978-1-4503-9238-9/22/06…$15.00
https://doi.org/10.1145/3512527.3531405

ACM Reference Format:
Yongbiao Chen, Sheng Zhang, Fangxin Liu, Zhigang Chang, Mang Ye, and Zhengwei Qi. 2022. TransHash: Transformer-based Hamming Hashing for Efficient Image Retrieval. In *Proceedings of the 2022 International Conference on Multimedia Retrieval (ICMR '22), June 27–30, 2022, Newark, NJ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3512527.3531405

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, and remote sensing systems. Among all the methods proposed for this challenging task [17, 18, 28, 45], hamming hash-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, 27, 54] learn their hash functions via the hand-crafted visual descriptors (e.g. *GIST* [46]). 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, which learns fine-grained features in an end-to-end fashion by complex deep neural networks, has become the dominant approach among the computer vision community [35–40]. Deep learning-based [15, 55] 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, 16, 21, 41] losses to preserve the similarity in the raw pixel space.

Recently, transformers [52] have demonstrated great successes in natural language processing [3, 12]. With the advent of the *Vision Transformer*, a variant of transformer tailored for computer vision

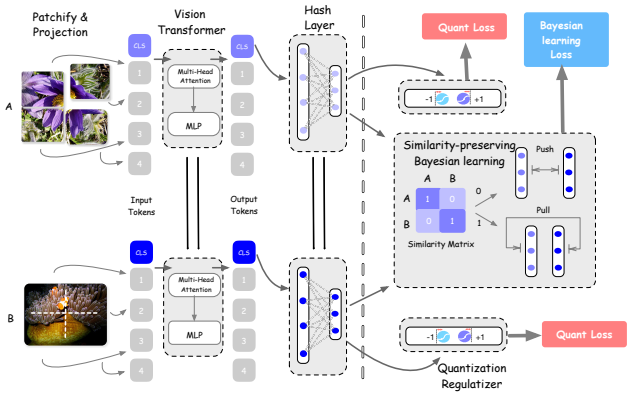

Figure 1: The brief architecture of our backbone Siamese vision transformer. For an image pair (A,B), we cut them into several patches. Then every patch is flattened and projected to a fix-sized embedding with a fully connected layer, resulting in a sequence of embeddings. Subsequently, we add a classification token in the front of each sequence. Then, two sequences are fed into the Siamese transformer architecture. 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 *Hamming* space for each pair.

tasks, transformers have trumped numerous CNN-based methods in various computer vision tasks (e.g. image classification [14], object re-identification [24], 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 Multi-Granular Vision Transformer backbone (**MGVT**), which comprises two identical multi-granular transformers sharing the same weight parameters[2]. Specifically, inspired by [24], 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. In parallel, we reorder the sequence of output features from the second last layer into K groups. The K groups are concatenated with the shared output token and then fed into another transformer layer to generate K local features. The primary merits are stated as follows. The model could simultaneously learn fine-grained global and local features with the joint global and local stream design. Secondly, our method could achieve similar effects as [32], which employs a divide-and-encode module to reduce the redundancy of the learned feature representation. Since the final learned representation is a concatenation of the global representation and several local representations, the subsets of the final feature vector are loosely correlated, resulting in increased independence and minimized redundancy. To further preserve the semantic similarity of the image pairs in the feature space, we propose to adopt the Bayesian learning framework to pull close similar pairs and push away dissimilar pairs in the embedding space for all the global and local features. Finally, since the learned feature representations are continuous in nature, we need to adopt the *sign* function $h = sign(f)$ to generate binary hamming hash code in the test stage. However, owing to the sizable gap between the continuous feature representation f and the hash code h after *sign* function, which is officially called the *quantization error* [62], directly generating hash codes with *sign* function in the testing stage could only lead to sub-optimal retrieval performances. In an effort to bridge the gap in the training stage, we reformulate the similarity-preserving learning problem as a constrained optimization problem. Concretely, on top of the Bayesian learning module, we add a Cauchy quantization loss [5] to statistically bridge the gap between the continuous feature representation and the binary hash coding.

To sum up, we make the following contributions:

(1) We innovate a siamese **M**ulti-**G**ranular **V**ision **T**ransformer (MGVT) backbone based on *ViT*. Specifically, 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 simultaneously. Meanwhile, as stated before, it could also promote the independence of the learned final hash code vector while reducing bit redundancy.
(2) By further adopting the similarity-preserving Bayesian learning module with a quantization constraint, we build up a novel deep hashing framework (**TransHash**) for large-scale image retrieval with a 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.
(3) 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 methods across three datasets by large margins.

2 RELATED WORKS

2.1 CNNs in Computer Vision

Convolutional neural network was first introduced in [33] 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* [30] that the CNN starts to become the workhorse of almost all the mainstream computer vision tasks, e.g. *Instance Segmentation* [1, 47],*Image Inpainting* [59, 60], *Deep hashing* [5, 6], *Person Re-identification* [10, 26, 57, 58] 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* [49], *GoogleNet* [50], *ResNet* [23], *EfficientNet* [51] 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

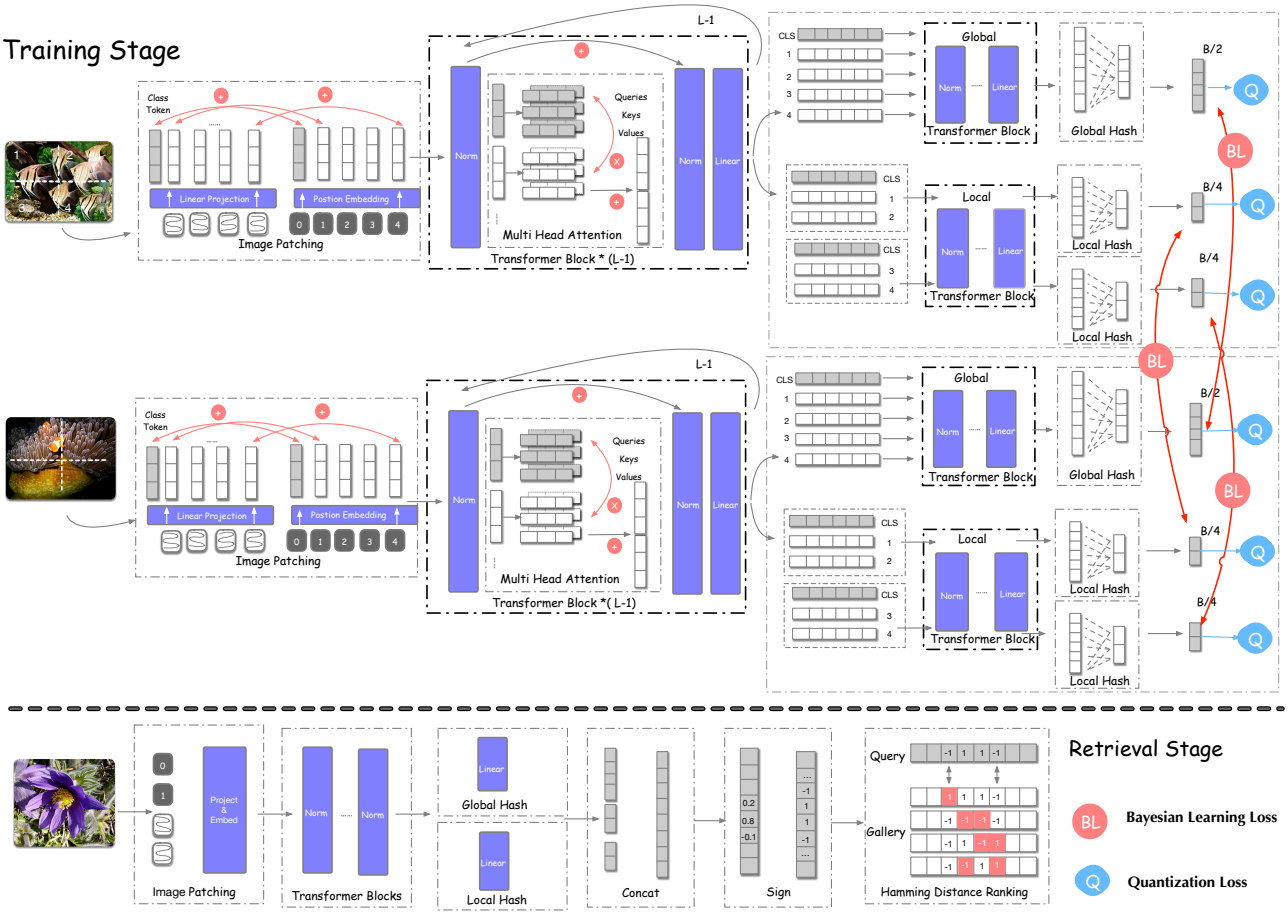

Figure 2: The detailed architecture of the proposed TransHash. The upper part denotes the training stage. Specifically, we follow the same protocols as ViT by feeding the patch embedding together with the position embedding into the transformer encoder. At the last layer of the transformer, we design two parallel transformer blocks: global and local transformer blocks. For the global feature and each local feature, we design a corresponding hashing layer. In the testing stage, the global and all the local hash vectors are concatenated and quantized into one hash code.

potent alternatives to convolutional neural networks. Our work is among the first endeavour to replace CNNs with pure transformer-based architectures in traditional computer vision tasks.

2.2 Transformer in Vision

The *Transformer* is first proposed in [52] for sequential data targeting 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, 19, 56]. In 2020, **Google** proposed *Vision Transformer* (ViT) [14], which applies a pure transformer directly to a sequence of image patches for image classification. Variants of *ViT* have achieved remarkable successes. For instance, [44] proposes a hierarchical vision transformer using shifted windows. [53] proposes a pyramid vision transformer tailored for dense prediction. Further, [24] 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 performance across multiple object re-identification datasets. *Vision transformer* is still in its nascent stages. Mounting research attention is being directed to investigating 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 [9, 18, 20, 25, 27, 28, 54]. 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) [27], which seeks to find a locality-sensitive hash family where the probability of hash collisions for similar objects is much higher than those for 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, 16, 34, 41, 62]. [34] proposes to learn the features and hash codes in an end-to-end manner. [62] 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. [61] innovates a new similarity matrix targeting multi-label image retrieval. [16] further introduces a deep polarized loss for the hamming code generation, obviating the need for an additional quantization loss.

3 PROPOSED METHOD

In this section, we will elaborate on the design of our framework.

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 $\mathbf{S}$ 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 $\mathcal{H} : \mathbf{I} \mapsto \{0, 1\}^B$ which encodes each input image I_i into a binary hash vector h_i with B bits while preserving the similarity information conveyed in $\mathbf{S}$. 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 $\{x_p^k\} \in \mathbb{R}^D, k \in [1, N]$. 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 z_0 is stated as follows:

$$z_0 = X_p + E_{pos} = [x_{class}; x_p^1, ..., x_p^N] + E_{pos} \tag{1}$$

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:

$$\begin{aligned} z_l &= \mathcal{F}_{msa}(\mathcal{F}_{ln}(z_{l-1})) + z_{l-1} \\ z_l &= \mathcal{F}_{mlp}(\mathcal{F}_{ln}(z_l)) + z_l \\ \text{where} \\ l &= 1...(L-1) \end{aligned} \tag{2}$$

Dual-stream feature learning. As proposed in [24], after the beforementioned self-attention encoder, we get the hidden features which are denoted as $Z_{L-1} = [z_{L-1}^0; z_{L-1}^1, z_{L-1}^2, ..., z_{L-1}^N]$. Note that, as stated before, z_{L-1}^0 is the hidden feature for the prepended learnable embedding x_{class}. Inspired by [24], 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 $Z_L = [f_g; z_L^1, z_L^2, ..., z_L^N]$, 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 fashion, K feature groups are derived which are denoted as$\{[z_{L-1}^0; z_{L-1}^1, ..., z_{L-1}^{N/K}], [z_{L-1}^0; z_{L-1}^{N/K+1}, ..., z_{L-1}^{2\times N/K}], [z_{L-1}^0; z_{L-1}^{N-N/K+1}, ..., z_{L-1}^N]\}$. 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

$$h_g = \mathcal{F}_h^g(f_g) = f_g W^T + b \tag{3}$$

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 $B/(2*K)$ output logits, resulting in K hash vectors $\{h_l^1, ..., h_l^K\}$.

In this way, for a image pair (I_i, I_j), the siamese model outputs two sets of hash vectors: $\{\{h_g\}^i, \{h_l^1\}^i, ..., \{h_l^K\}^i\}$ and $\{\{h_g\}^j, \{h_l^1\}^j, ..., \{h_l^K\}^j\}$, 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 $(I_i, I_j, s_{ij}) : s_{ij} \in \mathbf{S}$, where $s_{ij} = 1$ if I_i and I_j are from the same class and 0 otherwise, we can formulate the logarithm Maximum a Posteriori (**MAP**) estimation of the hash codes $\boldsymbol{H} = \{h_1, h_2, ..., h_P\}$ for P training points as:

$$\begin{aligned} &\log P(\boldsymbol{H} \mid \mathbf{S}) \propto \log P(\mathbf{S} \mid \boldsymbol{H}) P(\boldsymbol{H}) \\ &= \sum_{s_{ij} \in \mathbf{S}} w_{ij} \log P\left(s_{ij} \mid \boldsymbol{h}_i, \boldsymbol{h}_j\right) + \sum_{i=1}^{NT} \log P\left(\boldsymbol{h}_i\right) \end{aligned} \tag{4}$$

where $P(\mathbf{S} \mid \boldsymbol{H})$ is the weighted likelihood function and w_{ij} is the corresponding weight for each image pair (I_i, I_i). Since the similarity matrix $\mathbf{S}$ could be very sparse in real retrieval scenarios [6], it could lead to the data imbalance problem, resulting in sub-optimal

retrieval performances. The weighted likelihood is adopted to tackle this problem by assigning weights to each training pair according to the importance of misclassifying that pair [13]. To be clear, we set

$$w_{ij} = \begin{cases} |\mathrm{S}|/|\mathrm{S}_1|, & s_{ij} = 1 \\ |\mathrm{S}|/|\mathrm{S}_0|, & s_{ij} = 0 \end{cases} \tag{5}$$

where $\mathrm{S}_1 = \left\{s_{ij} \in \mathrm{S} : s_{ij} = 1\right\}$ is the set of similar pairs, $\mathrm{S}_0 = \left\{s_{ij} \in \mathrm{S} : s_{ij} = 0\right\}$ being the set of dissimilar pairs. For an pair h_i, h_j, $P\left(s_{ij} \mid \boldsymbol{h}_i, \boldsymbol{h}_j\right)$ is the conditional probability function of s_{ij} given a pair of hash codes h_i and h_j. Since the s_{ij} only takes two values 0 and 1, it is natural to define $P\left(s_{ij} \mid \boldsymbol{h}_i, \boldsymbol{h}_j\right)$ as a Bernoulli distribution:

$$\begin{aligned} P\left(s_{ij} \mid \boldsymbol{h}_i, \boldsymbol{h}_j\right) &= \begin{cases} \sigma\left(\mathcal{D}_H\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right), & s_{ij} = 1 \\ 1 - \sigma\left(\mathcal{D}_H\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right), & s_{ij} = 0 \end{cases} \\ &= \sigma\left(\mathcal{D}_H\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right)^{s_{ij}} \left(1 - \sigma\left(\mathcal{D}_H\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right)\right)^{1-s_{ij}} \end{aligned} \tag{6}$$

where $\mathcal{D}_H(.)$ is the Hamming distance function and σ 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 $\mathbf{h}_i \in \{-1, 1\}^B$ similar to [5, 6, 62]. Thus, we adopt a surrogate $\mathcal{D}_S$ for $\mathcal{D}_H$ in the continuous space which is formulated as:

$$\begin{aligned} \mathcal{D}_S\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) &= \frac{K}{4} \left\| \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|} - \frac{\boldsymbol{h}_j}{\|\boldsymbol{h}_j\|} \right\|_2^2 \\ &= \frac{K}{2}\left(1 - \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right) \end{aligned} \tag{7}$$

For the probability function σ, 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 $b/2$. 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:

$$\sigma\left(\mathcal{D}_S\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right) = \frac{\gamma}{\gamma + \mathcal{D}_S\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)} \tag{8}$$

where γ 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 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:

$$\begin{aligned} L_s &= \sum_{s_{ij} \in \mathbb{S}} L_{ce}(\boldsymbol{h}_i, \boldsymbol{h}_j) \\ &= \sum_{s_{ij} \in \mathrm{S}} w_{ij} \left(s_{ij} \log \frac{\mathcal{D}_s\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)}{\gamma} + \log\left(1 + \frac{\gamma}{\mathcal{D}_S\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)}\right)\right) \end{aligned} \tag{9}$$

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_i, I_j), we are increasing the value of $P(1|\mathbf{h}_i, \mathbf{h}_j)$, resulting in decreased value of $D_S(\mathbf{h}_i, \mathbf{h}_j)$ since σ 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 $P\left(\boldsymbol{h}_i\right) = \frac{\gamma}{\gamma + \mathcal{D}_S\left(|\boldsymbol{h}_i|, \mathbf{1}\right)}$ where γ is the same scale parameter as Eq. 8 and $\mathbf{1}$ is a vector of ones. Since we are maximizing $P(H)$ in Eq. 4, the quantization loss L_Q is stated as:

$$L_Q = \sum_{i=1}^{NT} Q(\boldsymbol{h}_i) = \sum_{i=1}^{NT} \log\left(1 + \frac{\mathcal{D}_S\left(|\boldsymbol{h}_i|, \mathbf{1}\right)}{\gamma}\right) \tag{10}$$

where $\mathbf{1}$ is a vector of ones. By minimizing the quantization loss Q in the training stage, each dimension of the hash vector $\mathbf{h}$ 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 $\{\{h_g\}^i, \{h_l^1\}^i, ..., \{h_l^K\}^i\}$ and $\{\{h_g\}^j, \{h_l^1\}^j, ..., \{h_l^K\}^j\}$ through the siamese vision transformer. Subsequently, for the local features, we obtain the Bayesian loss and quantization loss as:

$$\begin{aligned} L_B^{local} &= \sum_{s_{ij} \in \mathrm{S}} \sum_{k}^{K} L_{ce}(\{\mathbf{h}_l^k\}^i, \{\mathbf{h}_l^k\}^j) \\ L_Q^{local} &= \sum_{i}^{NT} \sum_{j}^{K} Q(\{h_l^j\}^i) \end{aligned} \tag{11}$$

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:

$$\min_{\theta} L_B^{global} + L_B^{local} + \lambda(L_Q^{global} + L_Q^{local}) \tag{12}$$

where θ denotes the set of parameters of the framework, and λ is the hyper-parameter for controlling the importance of the *Cauchy* quantization loss.

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 [29] is a dataset with 60,000 images from 10 classes. We follow the standard protocol in [5, 62]. 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 [11] 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, 62] 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.

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 [54] (NeurIPS)	-	-	-	-	0.4058	0.4209	0.4211	0.4104	0.2066	0.3280	0.3951	0.4191
ITQ[20] (TPAMI)	-	-	-	-	0.5086	0.5425	0.5580	0.5611	0.3255	0.4620	0.5170	0.5520
KSH[43] (CVPR)	-	-	-	-	0.3561	0.3327	0.3124	0.3368	0.1599	0.2976	0.3422	0.3943
BRE[31] (NeurIPS)	-	-	-	-	0.5027	0.5290	0.5475	0.5546	0.0628	0.2525	0.3300	0.3578
DSH[42] (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[62] (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[61] (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[16] (IJCAI)	0.825	0.838	0.830	0.829	-	-	-	-	0.684	0.740	0.756	0.756
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**

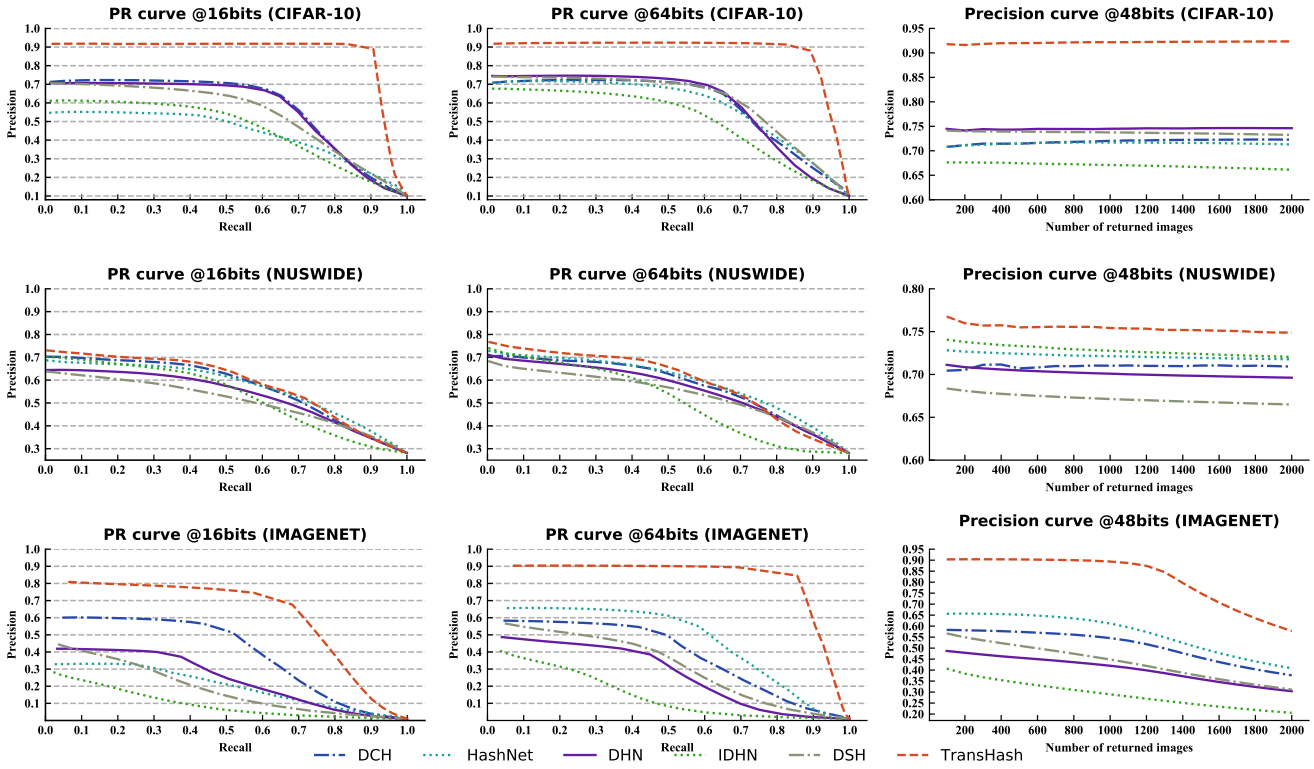

Figure 3: The experimental results of TransHash and other competing methods on three datasets

IMAGENET is a subset of the dataset for Large Scale Visual Recognition Challenge (ISVRC 2015) [48]. Specifically, we follow the same protocol as [16][6] by randomly sampling 100 classes and using 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** curves 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 Implementation Details

All the images are first resized to 256×256. 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.

4.3 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** [54], **ITQ** [20], **KSH** [43], and **BRE** [31] for detailed comparisons. For the deep learning-based hashing methods, we further include **DSH** [42] 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** [62], **HashNet** [6], **IDHN** [61] and **DPN** [16].

Note that, for all the non-deep methods and **DPN**, we directly quote the results from [6] and [16]. 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 224×224.

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 [42, 61]. **TransHash** tackles this problem by dynamically assigning a weight for each pair as is carried out in [6]. On **NUSWIDE**, our method 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.

4.4 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 σ, following the protocols in [62].

As shown in Tab. 2 and Fig. 4, when then 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 [62], we can see that the performance decreases sharply. Specifically, on **IMAGENET**, it experiences a conspicuous performance drop by an average of 5.55% **mAP**. We also note that the drop for shorter hash codes is more severe than for longer hash codes. The primary reason is that according to [5], the Cauchy distribution could effectively pull close similar pairs into a small Hamming radius, giving it an edge when the hash code length is short.

More importantly, to test the effectiveness of the proposed dual-stream feature learning, we also include the performances of the Siamese model with the solo global feature learning module. As depicted in Fig. 2, **TransHash w/o P** consistently underperform the model with dual-feature learning design. On **NUSWIDE** and **IMAGENET**, the average decline is 2.08% and 2.83%, respectively. The above experimental experiments have evidenced the effectiveness of the design of our pure transformer-based hashing framework.

Table 2: Mean Average Precision (MAP) of Different Variants of TransHash 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
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	0.8635
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

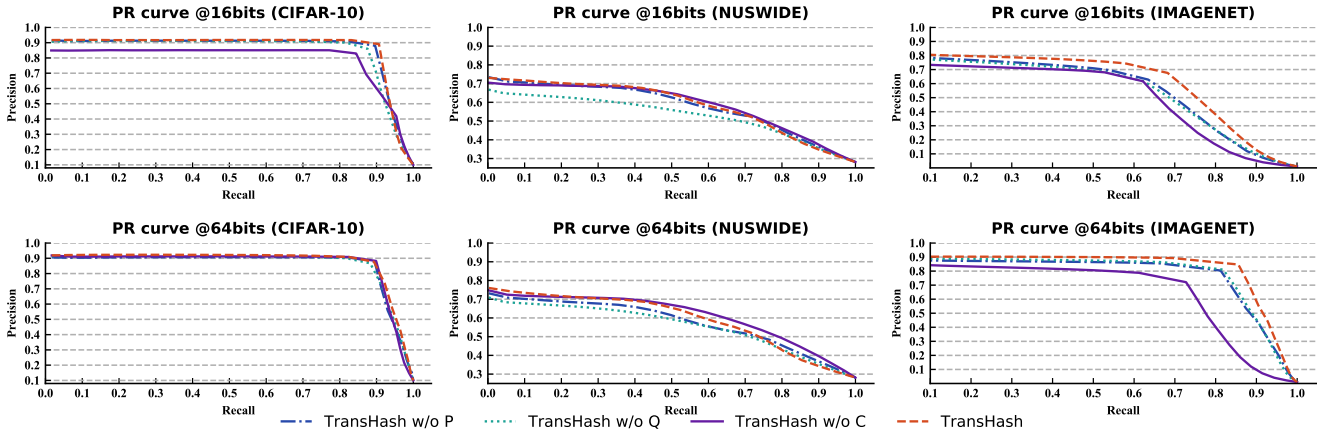

Figure 4: Experimental results of different variants of TransHash on three datasets

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	0.9057	0.9062	0.8994

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.

Since the hyper-parameter K, which controls how many groups we will divide our local features into, is rather important in our design, we further provide an ablation study on the sensitivity of K for various hash bits on **CIFAR-10**. Note that if the length of the final hash code vector is 16 and K equals 2, then the global feature is responsible for learning the first 8 bits and each local feature vector for the latter 4 bits.

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 along with 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.

ACKNOWLEDGMENTS

This work was supported in part by National NSF of China (NO. 62141218), and Shanghai Key Laboratory of Scalable Computing and Systems.

REFERENCES

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. 2019. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9157–9166.

[2] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* 7, 04 (1993), 669–688.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[4] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. 2019. Hashing with mutual information. *IEEE transactions on pattern analysis and machine intelligence* 41, 10 (2019), 2424–2437.

[5] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. 2018. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1229–1237.

[6] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*. 5608–5617.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 213–229.

[8] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 380–388.

[9] Yongbiao Chen, Sheng Zhang, Fangxin Liu, Chenggang Wu, Kaicheng Guo, and Zhengwei Qi. 2021. DVHN: A Deep Hashing Framework for Large-scale Vehicle Re-identification. *arXiv preprint arXiv:2112.04937* (2021).

[10] Yongbiao Chen, Sheng Zhang, and Zhengwei Qi. 2020. MAENet: Boosting Feature Representation for Cross-Modal Person Re-Identification with Pairwise Supervision. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*. 442–449.

[11] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*. 1–9.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Jacek P Dmochowski, Paul Sajda, and Lucas C Parra. 2010. Maximum Likelihood in Cost-Sensitive Learning: Model Specification, Approximations, and Upper Bounds. *Journal of Machine Learning Research* 11, 12 (2010).

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[15] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2475–2483.

[16] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. [n.d.]. Deep polarized network for supervised learning of accurate binary hashing codes. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. 825–831.

[17] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2017. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143* (2017).

[18] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2953.

[19] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. 2019. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 244–253.

[20] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2012. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE transactions on pattern analysis and machine intelligence* 35, 12 (2012), 2916–2929.

[21] Kun He, Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. 2018. Hashing as tie-aware learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4023–4032.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[24] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. 2021. TransReID: Transformer-based Object Re-Identification. *arXiv preprint arXiv:2102.04378* (2021).

[25] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. 2012. Spherical hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2957–2964.

[26] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[27] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. 1997. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 618–625.

[28] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 1097–1105.

[31] Brian Kulis and Trevor Darrell. 2009. Learning to Hash with Binary Reconstructive Embeddings.. In *NIPS*, Vol. 22. Citeseer, 1042–1050.

[32] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3270–3278.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[34] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2015. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855* (2015).

[35] Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, Tao Yang, and Li Jiang. 2021. SSTDP: Supervised Spike Timing Dependent Plasticity for Efficient Spiking Neural Network Training. *Frontiers in Neuroscience* 15 (2021).

[36] Fangxin Liu, Wenbo Zhao, Zhezhi He, et al. 2021. Bit-Transformer: Transforming Bit-level Sparsity into Higher Preformance in ReRAM-based Accelerator. In *Proceedings of the 40th International Conference on Computer-Aided Design*.

[37] Fangxin Liu, Wenbo Zhao, Zhezhi He, Yanzhi Wang, Zongwu Wang, Changzhi Dai, Xiaoyao Liang, and Li Jiang. 2021. Improving Neural Network Efficiency via Post-Training Quantization With Adaptive Floating-Point. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5281–5290.

[38] Fangxin Liu, Wenbo Zhao, Zongwu Wang, Tao Yang, and Li Jiang. 2021. IM3A: Boosting Deep Neural Network Efficiency via In-Memory Addressing-Assisted Acceleration. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*. 253–258.

[39] Fangxin Liu, Wenbo Zhao, Zongwu Wang, Yilong Zhao, Tao Yang, Yiran Chen, and Li Jiang. 2022. IVQ: In-Memory Acceleration of DNN Inference Exploiting Varied Quantization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[40] Fangxin Liu, Wenbo Zhao, Yilong Zhao, Zongwu Wang, Tao Yang, Zhezhi He, Naifeng Jing, Xiaoyao Liang, and Li Jiang. 2021. SME: ReRAM-based Sparse-Multiplication-Engine to Squeeze-Out Bit Sparsity of Neural Network. In *Proceedings of the 39th IEEE International Conference on Computer Design*.

[41] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2064–2072.

[42] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2064–2072.

[43] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2074–2081.

[44] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv preprint arXiv:2103.14030* (2021).

[45] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[46] Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision* 42, 3 (2001), 145–175.

[47] Mengye Ren and Richard S Zemel. 2017. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6656–6664.

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[49] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[51] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 6105–6114.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[53] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122* (2021).

[54] Yair Weiss, Antonio Torralba, Robert Fergus, et al. 2008. Spectral hashing.. In *Nips*, Vol. 1. Citeseer, 4.

[55] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.

[56] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. 2021. Segmenting transparent object in the wild with transformer. *arXiv preprint arXiv:2101.08461* (2021).

[57] Mang Ye, Xiangyuan Lan, Jiawei Li, and Pong Yuen. 2018. Hierarchical discriminative learning for visible thermal person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.