Vision Transformer Hashing for Image Retrieval

Shiv Ram Dubey, Satish Kumar Singh

Computer Vision and Biometrics Lab

Indian Institute of Information Technology, Allahabad, India

srdubey@iiita.ac.in, sk.singh@iiita.ac.in

Wei-Ta Chu

Department of Computer Science and Information Engineering

National Cheng Kung University, Taiwan

wtchu@gs.ncku.edu.tw

Abstract

Deep learning has shown a tremendous growth in hashing techniques for image retrieval. Recently, Transformer has emerged as a new architecture by utilizing self-attention without convolution. Transformer is also extended to Vision Transformer (ViT) for the visual recognition with a promising performance on ImageNet. In this paper, we propose a Vision Transformer based Hashing (VTS) for image retrieval. We utilize the pre-trained ViT on ImageNet as the backbone network and add the hashing head. The proposed VTS model is fine tuned for hashing under six different image retrieval frameworks, including Deep Supervised Hashing (DSH), HashNet, GreedyHash, Improved Deep Hashing Network (IDHN), Deep Polarized Network (DPN) and Central Similarity Quantization (CSQ) with their objective functions. We perform the extensive experiments on CIFAR10, ImageNet, NUS-Wide, and COCO datasets. The proposed VTS based image retrieval outperforms the recent state-of-the-art hashing techniques with a great margin. We also find the proposed VTS model as the backbone network is better than the existing networks, such as AlexNet and ResNet. The code is released at https://github.com/shivram1987/VisionTransformerHashing.

1. Introduction

Deep learning has witnessed a revolution for more than a decade for several applications of computer vision, natural language processing, and many more [19]. Deep learning learns the problem-specific important features automatically from data in an hierarchical fashion with different levels of abstractness. Different types of Neural Networks have been utilized to deal with different types of data. Specifically, Convolutional Neural Networks (CNNs) are used for image data [18]. Whereas, Recurrent Neural Networks (RNNs) are used for sequential data [30].

Several CNN models have been investigated for different computer vision applications, such as image classification [18], [14], object detection [10], [32], segmentation [13], [27], face recognition [35], and medical applications [28]. CNNs have also been very heavily exercised for image hashing and retrieval [7]. Deep Supervised Hashing (DSH) [26] is one of the early attempts to utilize CNN by quantizing the network outputs to binary hash code. The DSH uses a regularizer on the real-valued network outputs to produce the discrete binary values. Tanh function based continuation method is incorporated in HashNet [2] for smooth transition from real-valued features to binary codes. The HashNet also utilizes the weighted cross-entropy loss to learn the sparse data in a similarity-preserving manner. The gradients are passed as identity mapping for the hash layer to avoid the vanishing gradients in GreedyHash method [34]. The GreedyHash uses the Sign function in the hash layer. Improved Deep Hashing Network (IDHN) [41] utilizes the cross-entropy loss and mean square error loss to deal with 'hard similarity' and 'soft similarity', respectively, for multi-label image retrieval. Central Similarity Quantization (CSQ) [40] relies on the optimization of the central similarity between data points in terms of their hash centers to increase the distinctiveness of the hash codes for image retrieval. Deep Polarized Network (DPN) [9] utilizes polarization loss as bit-wise hinge-like loss which leads to the different output channels to be far away from zero. Inherently, DPN brings a high level of separability between the hash codes of different classes. These models utilize CNNs as the backbone network.

Though the CNNs have been very successful, it requires a deep network to achieve the global representation of the
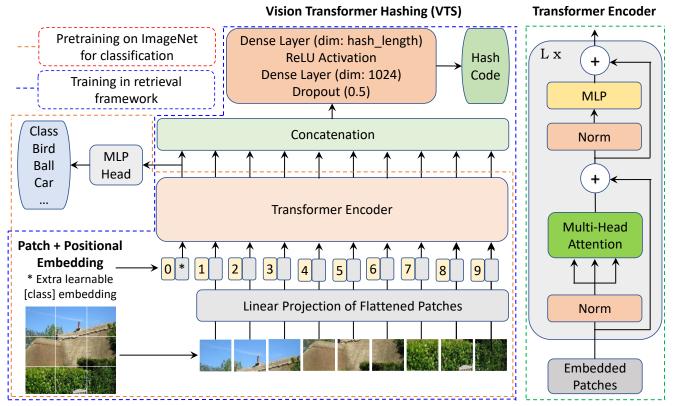
Figure 1. Vision Transformer Hashing (VTS) Framework for Image Retrieval in Blue Dotted Line using the Pre-trained Vision Transformer (ViT Model) [6] for Image Classification in Red Dotted Line.

image in feature space. However, at the other end, the RNNs are able to utilize the entire data in each step in the feature representation. But, the RNNs are not parallelizable as the input at a given time-step is dependent on the output of the previous time-step. Transformer networks are the recent trends in deep learning with very promising performance [37], [24]. The transformers tackle the downsides of both CNNs and RNNs by modeling the global representation of the input feature at each step with highly parallelizable architecture. Basically, the transformers utilize the self-attention modules [38]. The self-attention mechanism captures the relative importance within the same input sequence in the feature representation of that sequence.

The transformers have also witnessed its impact over computer vision problems [15], [12]. The Vision Transformer (ViT) [6] is the state-of-the-art to utilize the transformer for image recognition at scale. The ViT considers the patches of 16×16 of the image as the sequential input to the transformers. The ViT has shown outstanding performance as compared to the CNNs for visual recognition. Very recently, the vision transformers have gained a huge popularity for several vision tasks such as visual tracking [39], [3], dense prediction [31], medical image segmentation [21], person re-identification [22], text-to-visual retrieval [29], and human-object interaction detection [16],

among others.

A very few attempts have been made to use the transformer for image hashing [20], [11], [8], [36], [4]. Bidirectional transformer [20] utilizes the bidirectional correlations between frames for video hashing. However, the bidirectional transformer does not utilize the vision transformer. Transformer is used as an off-the-shelf feature extractor in [11]. Vision transformer is trained in [8] for image retrieval with a metric learning objective function utilizing the contrastive loss on real-valued descriptors. The hashing is not performed in [8]. Similarly, Reranking Transformer is used for image retrieval in [36] without hashing. TransHash [4] utilizes a siamese vision transformer for feature learning. However, the TransHash is not able to utilize the pre-training of Vision Transformers over large-scale dataset and the recently investigated improved objective functions for image hashing.

Motivated from the success of the transformers for different problems, we propose a Vision Transformer Hashing (VTS) for image retrieval in this paper. We utilize the pre-trained ViT [6] models to extract the real-valued features and fine-tune in several image hashing setup with additional hashing module. We use six state-of-the-art hashing frameworks with the proposed VTS model as the backbone network. An outstanding performance is observed using the

2

proposed VTS model on several benchmark image retrieval datasets. The contributions of are as follows:

- We propose an end-to-end trainable Vision Transformer Hashing (VTS) model for image retrieval by utilizing the pre-trained ViT models [6].

- The proposed VTS model utilizes the ViT model as a generic feature extractor, removes the MLP head, appends a hashing module and uses the existing hashing frameworks to train the model.

- Extensive experiments have been conducted to show the excellent performance by the VTS model on benchmark CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets under six state-of-the-art hashing frameworks.

The remaining paper is structured as follows: Section 2 introduces the Vision Transformer Hashing (VTS); Section 3 presents the experimental setup; Section 4 discusses the results; and finally Section 5 sets the concluding remarks.

2. Proposed Vision Transformer Hashing

The proposed Vision Transformer Hashing (VTS) framework is illustrated in Fig. 1. It consists of patch embedding generation, transformer encoder, pre-training under classification framework and hashing module for image retrieval.

2.1. Patch Embeddings Generation

The input image $I \in \mathbb{R}^{m,m,c}$ is divided into N non-overlapping patches $I_i \in \mathbb{R}^{k,k,c}$ where $i = 1, 2, ..., N$, $m^2 = Nk^2$ and $c = 3$ is the number of color channels. The input patches I_i are flattened into vectors $V_i \in \mathbb{R}^{1,d}$ where $d = ck^2$ and $i = 1, 2, ..., N$. A linear projection is performed on each flattened vector to generate the patch embeddings (PE) as,

$$PE_i = V_i \times W_{PE} \tag{1}$$

where $V_i \in \mathbb{R}^{1,d}$ is the flattened vector corresponding to the ith patch, $W_{PE} \in \mathbb{R}^{d,de}$ is the parameter matrix and $PE_i \in \mathbb{R}^{1,de}$ is the projected embeddings corresponding to the ith patch. Here, de is the embedding dimension (i.e., hidden size) of the projection. Thus, the projected embeddings corresponding to the entire input image can be written as $PE \in \mathbb{R}^{N,de}$. A class token embedding $CT \in \mathbb{R}^{1,de}$ is introduced as the learnable parameters initialized with zeros and concatenated with the projected embeddings in order to generate the expanded embeddings ($EE \in \mathbb{R}^{N+1,de}$) as $EE = [CT, PE]$ on 1st dimension. In order to incorporate the spatial information, the position embedding ($PoE \in \mathbb{R}^{N+1,de}$) is included as the learnable parameters

initialized with zeros and added in the expanded embeddings. So, the final projection embedding (FE) is given as,

$$FE = dropout(EE + PoE, 0.1) \tag{2}$$

where $dropout$ is a dropout layer with 0.1 dropout factor and $FE \in \mathbb{R}^{N+1,de}$ is the final projection embeddings to be given as the input to transformer encoder.

2.2. Transformer Encoder

The transformer encoder consists of the stack of L transformer blocks. A transformer block transforms the given hidden-state input into an output of the same dimension with the help of a self-attention mechanism. The input to a transformer block $T_j|_{j=1,2,...,L}$ is the hidden state h_{j-1} which is the output of previous transformer block T_{j-1}, where $h_{j-1} \in \mathbb{R}^{N+1,de}$. The input to the first transformer block T_1 is the final projection embedding (i.e., $h_0 = FE$). The output of the last transformer block T_L is hidden state h_L. Basically, the input and output of the jth transformer block are denoted by h_{j-1} and h_j, respectively.

The layer normalization is the first layer in the jth transformer block which takes input h_{j-1} and produces output $l_{j-1} \in \mathbb{R}^{N+1,de}$. The output of layer normalization is the input to self-attention layer which consists of three parametric projections for Query, Key and Value vectors using the learnable weight matrices W_Q, W_K and W_V, respectively. The output of linear projections are

$$S_t = l_{j-1} \times W_t \tag{3}$$

where $S_t \in \mathbb{R}^{N+1,de}$ for $t = \{Q, K, V\}$. As the attention layer uses multi-attention heads, the Query (S_Q) and Value (S_V) vectors are reshaped to a dimension of $(A_h, N+1, A_S)$ and the Key vector is reshaped to the dimension of $(A_h, A_S, N+1)$ where A_h is the number of attention heads and A_S is the attention head size which is computed as $A_S = de/A_h$. Next, a matrix multiplication is performed between S_Q and S_K to produce an output matrix $S_{QK} \in \mathbb{R}^{A_h,N+1,N+1}$ which is normalized by square root of attention head size as,

$$S_{QK} = \frac{S_Q \times S_K}{\sqrt{A_S}}. \tag{4}$$

Next, the attention weights (A_w) are computed using softmax function on S_{QK} as,

$$A_w = softmax(S_{QK}). \tag{5}$$

The Value vector (S_V) is operated with the attention weights (A_w) to generate the attention weighted features ($F_A \in \mathbb{R}^{A_h,N+1,A_S}$) as,

$$F_A = A_w \times S_V \tag{6}$$

3

which is further reshaped in the dimension of $(N + 1, de)$, i.e., $F_A \in \mathbb{R}^{N+1,de}$. Finally, a linear projection is applied on F_A with a learnable parameter W_A to produce the output of the self-attention module (F_S) as $F_S = F_A \times W_A$. In order to improve the training, the transformer blocks include the residual connection as,

$$F_R = h_{j-1} + F_S \qquad (7)$$

where h_{j-1} is the input to transformer block, F_S is the output of self-attention module and F_R is the output of residual connection. A layer normalization is then used on the output of residual connection F_R with the output of this layer as l_R of same dimensionality. A multilayer perceptron module is applied on the output of layer normalization with a linear projection to $mlp_{dim} = 3072$ dimension, GELU activation function, dropout layer with 0.1 dropout factor and a linear projection to $de = 768$ dimension. Finally, a residual connection is used to produce the output of the transformer block as,

$$h_j = F_R + F_{MLP} \qquad (8)$$

where $F_{MLP} \in \mathbb{R}^{N+1,de}$ is the output of the MLP module, F_R is the output of the first residual connection and h_j is the output of the jth transformer block. Similarly, the output of the last transformer block h_L is also computed. The final output of the transformer encoder (i.e., $TE \in \mathbb{R}^{N+1,de}$) is the layer normalization output on h_L which is passed to the next step, such as pre-training under classification framework and hashing under retrieval framework (see Fig. 1).

2.3. Vision Transformer Pretraining

In this paper, we use the pretrained Vision Transformer (ViT) model [6]. The pretraining is performed using the final output of the transformer encoder on the ImageNet dataset under classification framework. The ViT extracts the representative features $RF \in \mathbb{R}^{1,de}$ from the first vector of transformer encoder output TS. The ViT uses a multilayer perceptron (MLP) head which computes the class scores from the RF features. We use two versions of pretrained ViT, including ViT-B_32 and ViT-B_16 which use the patch sizes of 32 and 16, respectively. We use the code and pretrained models available from GitHub open repository[1] for hashing.

2.4. Hashing Module

We train the pre-trained vision transformer model under image retrieval framework. In order to facilitate the image retrieval and learning of hash code, we add a hash block on top of the transformer encoder output. First, the output of transformer encoder TE is reshaped to a dimension of $((N + 1) \times de)$ which is passed through a dropout layer with 0.5 dropout factor. Then a linear projection is applied

to transform the feature into a dimension of 1024 followed by the ReLU activation function. Finally, a linear projection is applied to generate the final hash features (H_f) having the number of values the same as the hash bit length. Note that we use six state-of-the-art methods to generate the hash code from hash features. Based on the input patch size, i.e., 32 and 16, we propose two VTS variants, i.e., VTS32 and VTS16, respectively.

In this paper, we test the VTS performance under six state-of-the-art hashing frameworks, namely Deep Supervised Hashing (DSH) [26], HashNet [2], GreedyHash [34], Improved Deep Hashing Network (IDHN) [41], Central Similarity Quantization (CSQ) [40] and Deep Polarized Network (DPN) [9]. We use the hash features H_f as the input to these frameworks and train the entire network including transformer encoder and hash module using the objective function of the corresponding hashing framework. Basically, the vision transformer with hash module (i.e., VTS) can be seen as the backbone network under these hashing frameworks. The code for these hashing frameworks is considered from the open GitHub repository[2]. The experiments are also carried out using AlexNet [18] and ResNet50 [14] backbone networks to show the improved performance using the proposed VTS32 and VTS16 backbone networks.

3. Experimental Setup

This section describes the experimental setup, including datasets used for the retrieval experiments, evaluation metrics, network and training settings.

3.1. Datasets Used

We use four datasets which are widely adapted for image retrieval, including CIFAR10, ImageNet, NUS-Wide and MS-COCO. The CIFAR10 dataset [17] is a collection of $60,000$ images from 10 categories with $6,000$ images per category. We follow the standard experimental practice on the CIFAR10 dataset. Similar to [42], [1], [4], we randomly sample $5,000$ images for the training set with 500 images from each category. Then, the query set having 1000 images is sampled randomly with 100 images per category. The rest of the images are used as the database. This is referred to as CIFAR10@54000. We also compute the results on the CIFAR10 dataset by following the protocol of [9] which considers the training set also as part of the database. This is referred to as CIFAR10@All. ImageNet dataset is a subset of the Large Scale Visual Recognition Challenge (ILSVRC 2015) [33]. We use the standard retrieval protocol [40], [9] on the ImageNet dataset. Basically, 100 classes are randomly sampled. All the validation set images of these classes are used in the query set. However, all the training set images of these classes are used as the database.

---

[1]https://github.com/jeonsworld/ViT-pytorch

[2]https://github.com/swuxyj/DeepHash-pytorch

4

Table 1. The retrieval results in terms of mAP in % for 16, 32 and 64 bit hash codes using AlexNet, ResNet and VTS backbone networks under DSH [26], HashNet [2], GreedyHash [34], IDHN [41], CSQ [40] and DPN [9] frameworks on CIFAR-10, ImageNet, NUS-Wide and MS-COCO datasets. The best results are highlighted in bold.

Backbone	DSH			HashNet			GreedyHash			IDHN			CSQ			DPN		
Network	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b
Results on CIFAR-10 dataset (mAP@54000 in %)																		
AlexNet	73.5	77.7	78.5	65.1	77.7	79.8	76.1	78.0	81.6	75.9	76.4	76.7	75.9	78.0	78.3	73.5	76.1	77.9
ResNet	69.1	67.0	52.4	55.5	86.8	88.0	81.5	83.5	86.3	75.6	85.6	86.5	83.6	82.8	83.7	82.0	83.8	82.0
VTS32	93.0	94.6	95.3	96.2	97.1	95.6	94.8	95.6	94.3	95.9	95.8	96.0	96.0	96.1	95.2	95.1	95.6	95.0
VTS16	**98.0**	**97.4**	**97.4**	**97.8**	**98.2**	**98.3**	**97.0**	**97.0**	**98.0**	**97.6**	**97.5**	**98.4**	**97.9**	**97.9**	**97.6**	**97.4**	**96.7**	**97.5**
Results on CIFAR-10 dataset (mAP@All in %)																		
AlexNet	76.0	78.5	79.0	65.9	78.0	80.1	76.9	81.8	81.1	76.4	77.5	78.1	79.6	78.3	76.8	76.2	79.8	79.2
ResNet	64.3	57.5	56.3	58.9	86.1	85.7	82.2	85.8	87.0	85.5	85.9	85.6	84.9	83.5	85.1	83.7	83.3	84.0
VTS32	93.7	95.0	95.4	96.2	95.9	96.6	95.6	94.9	95.8	95.5	95.5	94.3	96.2	97.0	96.1	96.3	95.5	95.2
VTS16	**97.4**	**98.1**	**98.3**	**98.1**	**97.9**	**98.2**	**97.0**	**97.7**	**97.5**	**97.8**	**98.0**	**97.9**	**97.6**	**97.7**	**97.6**	**96.8**	**96.5**	**97.7**
Results on ImageNet dataset (mAP@1000 in %)																		
AlexNet	50.4	60.6	64.4	35.9	57.2	64.8	57.7	65.8	68.2	02.0	37.7	49.9	60.5	67.5	69.6	59.2	67.8	69.2
ResNet	47.3	67.3	75.3	20.5	47.8	66.0	81.2	84.6	85.6	04.3	36.5	58.0	83.6	86.0	86.6	83.3	85.7	86.6
VTS32	51.8	80.6	86.0	61.1	81.2	84.4	86.5	86.5	88.0	71.0	**72.7**	**66.8**	86.6	88.3	88.6	84.5	88.1	88.7
VTS16	**85.6**	**90.8**	**91.7**	**81.7**	**88.3**	**89.5**	**90.3**	**91.1**	**91.6**	**80.5**	67.2	64.2	**90.6**	**91.5**	**91.8**	**90.2**	**91.4**	**91.8**
Results on NUS-Wide dataset (mAP@5000 in %)																		
AlexNet	77.5	79.2	80.8	75.2	81.6	84.5	74.8	**78.8**	80.0	79.1	80.6	81.3	78.0	82.1	83.4	75.0	81.3	83.5
ResNet	77.1	79.7	79.1	76.8	82.3	83.9	75.6	77.7	78.8	80.9	79.9	79.4	79.3	83.6	83.8	78.9	82.5	84.0
VTS32	79.6	80.5	**82.9**	77.6	83.8	86.2	75.8	78.2	79.7	83.3	84.1	84.1	**82.4**	**85.5**	**86.1**	80.2	84.1	**86.1**
VTS16	**79.8**	**81.7**	**82.9**	**79.2**	**85.2**	**87.3**	**76.2**	76.9	78.6	**84.1**	**85.2**	**85.1**	81.9	84.6	85.3	79.7	83.5	84.6
Results on MS-COCO dataset (mAP@5000 in %)																		
AlexNet	63.5	65.2	67.5	63.3	68.9	73.3	63.4	69.7	72.7	65.6	68.7	69.9	63.0	72.0	76.6	63.3	72.3	76.8
ResNet	69.0	72.3	75.5	64.4	68.1	72.8	69.5	75.1	77.7	71.3	68.6	58.6	71.2	81.8	87.2	71.8	80.9	85.2
VTS32	71.9	75.8	76.9	70.6	76.8	80.6	75.5	79.6	80.3	76.3	79.9	78.4	77.0	86.5	88.9	75.3	84.5	88.0
VTS16	**80.5**	**84.3**	**85.0**	**75.2**	**82.8**	**86.5**	**78.0**	**81.2**	**82.3**	**77.8**	**82.8**	**83.0**	**77.9**	**87.4**	**91.1**	**78.5**	**85.8**	**89.7**

From the database images, $13,000$ images are sampled as the training set. NUS-Wide dataset is one of the widely used dataset for image retrieval [5]. We follow the protocol used in [9] where 21 most frequent concepts are considered as image annotations leading to $195K$ images. The query set is sampled randomly with 100 images per concept. The remaining images are considered as a database for retrieval. The training set consists of 500 images per concept randomly sampled from the database. MS-COCO dataset consists of images from 80 categories [25]. We use the existing setup [1], [40] of MS-COCO dataset for image retrieval which includes $5,000$, $10,000$ and $1,17,218$ images in query, training and retrieval sets, respectively.

3.2. Evaluation Metrics

We follow the standard practice [40], [9], [4] of evaluating the image retrieval methods and compute the Mean Average Precision (mAP). The mAP is computed for $54,000$, $59,000$, $1,000$, $5,000$ and $5,000$ retrieved images on CIFAR10@54000, CIFAR10@All, ImageNet, NUS-Wide and MS-COCO datasets, respectively. Precision vs Recall curve is also used to observe the ROC characteristics.

3.3. Network and Training Settings

In the experiments, all the images are resized with $m = 224$. Two variants of the proposed Vision Transformer Hashing (VTS) are used for experiments, i.e., VTS32 and VTS16 having patch size $k = 32$ and 16, respectively. Thus, the number of patches $N = 49$ and 196 for VTS32 and VTS16, respectively. The hidden size (de) is set as 768 in the experiments. The number of transformer blocks (L) as well as the number of attention heads (A_h) are 12 in the VTS models. The hash code is generated with 16, 32 and 64 bit length. All the models are trained for 150 epochs with a batch size of 32 using Adam optimizer with a learning rate of $1e^{-5}$. The testing is performed at every 30 epochs and the best results are reported.

4. Experimental Results and Analysis

This section presents the detailed experimental results over four benchmark datasets using the proposed VTS models under six retrieval frameworks. The ROC plots are also
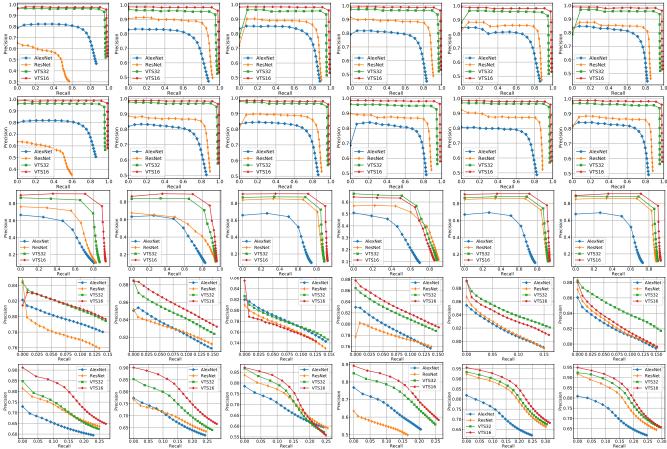
Figure 2. The RoC curves by using 64 bit hash code generated from AlexNet, ResNet, VTS32 and VTS16 backbone networks under DSH (1st column), HashNet (2nd column), GreedyHash (3rd column), IDHN (4th column), CSQ (5th column), and DPN (6th column) frameworks on CIFAR-10@54000 (1st row), CIFAR-10@All (2nd row), ImageNet@1000 (3rd row), NUS-Wide@5000 (4th row), and MS-COCO@5000 (5th row) datasets.

presented. The results are also compared with the state-of-the-art methods. An ablation study is also performed on the utilization of the features of the transformer encoder. We perform the experiments using PyTorch deep learning library on a system with 24 GB NVIDIA RTX GPU.

4.1. Experimental Results

The experimental results in terms of mAP (%) are summarized in Table 1 for image retrieval on CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets. The results of the proposed VTS32 and VTS16 backbone networks are compared with the AlexNet and ResNet50 backbone networks. The results are computed using 16 bit (16b), 32 bit (32b) and 64 bit (64b) hash codes under six retrieval frameworks, namely DSH, HashNet, GreedyHash, IDHN, CSQ and DPN. It can be noted that the proposed VTS16 outperforms the other backbone networks in all the settings on CIFAR10 and MS-COCO datasets and also better in most of the cases on ImageNet and NUS-Wide datasets. Moreover, the performance of VTS32 is also superior to AlexNet

and ResNet50 in the majority of cases. Following are the observations:

- On an average the mAP of VTS16 backbone network is improved by 26.65%, 54.68%, 3.15%, and 20.56% as compared to AlexNet backbone network and by 23.74%, 31.50%, 2.67%, and 12.78% as compared to ResNet50 backbone network on CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets, respectively.

- On an average the mAP of VTS32 backbone network is improved by 23.86%, 42.94%, 3.03%, and 15.14% as compared to AlexNet backbone network and by 21.01%, 21.51%, 2.56%, and 7.71% as compared to ResNet50 backbone network on CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets, respectively.

We also compare the precision recall ROC plots of AlexNet, ResNet50, VTS32 and VTS16 backbone networks for 64 bit hash code in Fig. 2. The ROC plots corresponding to DSH, HashNet, GreedyHash, IDHN, CSQ and DPN re-

Table 2. The retrieval results comparison in terms of mAP (%) with state-of-the-art results on CIFAR-10, ImageNet, NUS-Wide and MS-COCO datasets for 16, 32 and 64 bit hash codes. The results of the proposed VTS16 model under CSQ framework are used for comparison. The $\star$, $\dagger$ and $\ddagger$ symbols indicate that the corresponding results are taken from DCH [1], DFH [23] and DPN [9], respectively. The $+$ symbol indicates that all 81 classes of NUS-Wide dataset are used for these results. The best results are highlighted in bold.

Method	CIFAR10@54000			CIFAR10@All			ImageNet@1000			NUS-Wide@5000			MS-COCO@5000		
	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b
DSH (CVPR) [26]	-	-	-	-	66.1		-	-	-	-	55.8	-	-	-	-
DHN (AAAI) [42]	69.3*	64.5*	58.8*	-	-	-	31.1†	47.2†	57.3†	-	74.8	-	67.5*	66.8*	41.9*
HashNet (ICCV) [2]	74.8*	77.8*	62.6*	64.3‡	67.5‡	68.7‡	50.6	63.1	68.4	66.2$^+$	69.9$^+$	71.6$^+$	68.7	71.8	73.6
GreedyHash (NIPS) [34]	-	-	-	-	81.0	-	62.5	66.2	68.8	-	-	-	-	-	-
DCH (CVPR) [1]	79.0	79.8	79.4	-	-	-	-	-	-	74.0$^+$	77.2$^+$	71.2$^+$	70.1	75.8	70.1
DFH (BMVC) [23]	-	-	-	-	83.1	-	59.0	69.7	74.7	-	83.3	-	-	-	-
CSQ (CVPR) [40]	-	-	-	-	-	-	85.1	86.5	87.3	81.0	82.5	83.9	**79.6**	83.8	86.1
DPN (IJCAI) [9]	-	-	-	77.4	80.3	81.2	60.6	69.3	72.9	81.0	83.2	83.9	-	-	-
TransHash [4]	90.8	91.1	91.7	-	-	-	78.5	87.3	89.2	72.6$^+$	73.9$^+$	74.9$^+$	-	-	-
VTS16-CSQ (Proposed)	**97.9**	**97.9**	**97.6**	**97.6**	**97.7**	**97.6**	**90.6**	**91.5**	**91.8**	**81.9**	**84.6**	**85.3**	77.9	**87.4**	**91.1**

Table 3. The retrieval results in terms of mAP in % for 16, 32 and 64 bit hash codes using only cls features and all features of the proposed VTS32 and VTS16 models under DSH, HashNet, GreedyHash, IDHN, CSQ and DPN frameworks on CIFAR-10, ImageNet, NUS-Wide and MS-COCO datasets. The best results are highlighted in bold.

Backbone Network	DSH			HashNet			GreedyHash			IDHN			CSQ			DPN		
	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b	16b	32b	64b
Results on CIFAR-10 dataset (mAP@54000 in %)																		
VTS32 (cls)	**93.2**	**94.6**	95.1	95.6	96.4	95.5	94.2	93.7	**94.8**	94.8	94.8	94.6	95.2	95.1	**95.6**	**95.8**	95.0	93.8
VTS32 (all)	93.0	**94.6**	**95.3**	96.2	97.1	95.6	**94.8**	**95.6**	94.3	**95.9**	95.8	**96.0**	**96.0**	**96.1**	95.2	95.1	**95.6**	**95.0**
VTS16 (cls)	96.2	96.9	97.0	**97.8**	97.3	97.9	94.8	95.9	97.2	97.1	97.1	97.1	**97.8**	**97.6**	97.1	96.8	96.2	96.4
VTS16 (all)	**98.0**	**97.4**	**97.4**	97.8	**98.2**	**98.3**	**97.0**	**97.0**	**98.0**	**97.6**	**97.5**	**98.4**	97.9	**97.9**	**97.6**	**97.4**	96.7	97.5
Results on CIFAR-10 dataset (mAP@All in %)																		
VTS32 (cls)	92.7	**96.3**	**96.2**	95.6	95.8	95.9	95.4	**96.0**	96.3	95.0	**95.7**	95.4	96.1	95.4	**96.7**	94.2	94.1	92.9
VTS32 (all)	**93.7**	95.0	95.4	96.2	95.9	96.6	95.6	94.9	95.8	**95.5**	95.5	94.3	**96.2**	**97.0**	96.1	**96.3**	**95.5**	**95.2**
VTS16 (cls)	97.0	97.8	97.5	97.5	97.7	**98.3**	95.5	96.7	96.4	97.7	97.3	97.3	**97.9**	97.1	**97.6**	96.1	96.1	95.6
VTS16 (all)	**97.4**	**98.1**	**98.3**	98.1	**97.9**	98.2	**97.0**	**97.7**	**97.5**	**97.8**	**98.0**	**97.9**	97.6	**97.7**	97.6	**96.8**	**96.5**	**97.7**
Results on ImageNet dataset (mAP@1000 in %)																		
VTS32 (cls)	**77.5**	**85.5**	**88.0**	80.7	83.1	85.3	84.9	**87.0**	86.6	**80.6**	**79.6**	71.3	84.2	86.4	86.7	**84.9**	87.1	86.7
VTS32 (all)	51.8	80.6	86.0	61.1	81.2	84.4	**86.5**	86.5	**88.0**	71.0	72.7	66.8	**86.6**	**88.3**	**88.6**	84.5	**88.1**	**88.7**
VTS16 (cls)	81.0	90.1	91.1	**87.2**	**88.4**	89.3	89.4	90.3	90.2	**88.6**	**84.4**	**79.2**	89.9	89.1	89.0	88.6	90.6	90.4
VTS16 (all)	**85.6**	**90.8**	**91.7**	81.7	88.3	**89.5**	90.3	91.1	91.6	80.5	67.2	64.2	**90.6**	**91.5**	**91.8**	90.2	91.4	91.8
Results on NUS-Wide dataset (mAP@5000 in %)																		
VTS32 (cls)	77.1	**81.2**	81.4	78.5	83.6	**86.3**	76.5	78.4	80.6	**83.7**	**84.4**	83.9	80.9	84.9	**86.2**	80.4	83.2	**86.3**
VTS32 (all)	**79.6**	80.5	**82.9**	77.6	**83.8**	86.2	75.8	78.2	79.7	83.3	**84.1**	84.1	**82.4**	**85.5**	86.1	80.2	**84.1**	86.1
VTS16 (cls)	**80.1**	81.0	81.4	79.1	85.0	87.2	76.2	76.9	78.5	83.9	85.0	83.8	79.9	83.9	**85.3**	77.0	83.2	**85.3**
VTS16 (all)	79.8	**81.7**	**82.9**	79.2	**85.2**	**87.3**	76.2	76.9	**78.6**	**84.1**	**85.2**	**85.1**	81.9	84.6	**85.3**	**79.7**	**83.5**	84.6
Results on MS-COCO dataset (mAP@5000 in %)																		
VTS32 (cls)	**73.0**	75.8	77.9	70.7	75.8	79.9	73.7	77.2	79.2	75.3	**81.1**	78.3	70.2	83.0	85.9	71.3	82.7	85.8
VTS32 (all)	71.9	**75.8**	76.9	70.6	**76.8**	**80.6**	75.5	79.6	80.3	**76.3**	79.9	**78.4**	**77.0**	**86.5**	**88.9**	75.3	84.5	88.0
VTS16 (cls)	78.6	81.8	84.3	75.1	82.7	84.2	77.0	79.9	80.7	77.4	**83.8**	69.3	72.5	83.0	86.5	73.5	78.6	86.6
VTS16 (all)	**80.5**	**84.3**	**85.0**	75.2	82.8	**86.5**	**78.0**	**81.2**	**82.3**	**77.8**	82.8	**83.0**	77.9	**87.4**	**91.1**	78.5	85.8	89.7

trieval frameworks are drawn in 1st to 6th columns, respectively. The 1st to 5th rows correspond to ROC plots on CIFAR10@54000, CIFAR10@All, ImageNet@1000, NUS-Wide@5000 and MS-COCO@5000 datasets in order. It is evident from these plots that both the VTS16 and VTS32 backbone networks perform better than the AlexNet and ResNet50 backbone networks in all the cases except on

NUS-Wide under the GreedyHash framework. It is also observed that VTS16 is superior in the majority of cases.

4.2. Comparison with Existing Results

Table 2 presents the results of the proposed vision transformer hashing with 16 patch size under CSQ retrieval framework (i.e., VTS16-CSQ) and the results of the state-

of-the-art retrieval models, including DSH [26], DHN [42], HashNet [2], GreedyHash [34], DCH [1], DFH [23], CSQ [40], DPN [9] and TransHash [4]. Note that TransHash is a recent siamese transformer model for image retrieval. Following are the observations from these results comparison:

- It is observed that the proposed hashing technique outperforms the existing hashing approaches on CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets for 16 bit, 32 bit and 64 bit hash codes.

- It is also evident that the pre-training of vision transformer models plays an important role as the VTS16-CSQ model clearly swaps the TransHash model on CIFAR10, ImageNet and NUS-Wide datasets.

4.3. Ablation Study

We use *all* features (TE) of output of the transformer encoder in the hash layer in the proposed VTS32 and VTS16 models in the earlier results. In order to analyze the impact of features, the results using only class (*cls*) features of the proposed transformer model are compared with all the features in Table 3 as an ablation study under different retrieval frameworks. Note that the *cls* feature represents the output features of the vision transformer corresponding to the output unit which was used for MLP Head originally as shown in Fig. 1. The VTS32 (cls) and VTS32 (all) refer to the computed hash code from only *cls* features and *all* features, respectively, while the patch-size is considered as 32. Similarly, the VTS16 (cls) and VTS16 (all) refer to the computed hash code from only *cls* features and *all* features, respectively, while the patch-size is considered as 16. It can be observed that the performance of VTS16 is superior in most of the cases with *all* features. However, the performance of VTS32 using *cls* features is generally better on ImageNet dataset as the pretrained vision transformer model was originally trained on the same dataset. Overall, the utilization of *all* features to compute the VTS hash code is advised in order to improve the generalization capability of the network.

5. Conclusion

In this paper, we propose a vision transformer based hashing (VTS) framework for image retrieval using the pre-trained ViT model as the backbone network. We train the VTS models under six retrieval frameworks to extract the hash code. The training is performed in an end-to-end fashion using the objective function of the corresponding retrieval framework. The results are compared on CIFAR10, ImageNet, NUS-Wide and MS-COCO datasets. It is noticed that the proposed VTS backbone outperforms the AlexNet and ResNet backbones under different retrieval frameworks. Moreover, the performance of the proposed

model is superior to the state-of-the-art hashing techniques, including transformer based technique. We also observe that the CSQ framework is better suited with the proposed VTS model for image retrieval. It is also found that the consideration of all the features of the vision transformer's output increases the generalization capability of the model. Thus, it can be concluded that the vision transformer based hashing significantly boosts the performance of image retrieval due to it's learning capability of discriminative features.

References

[1] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018. 4, 5, 7, 8

[2] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017. 1, 4, 5, 7, 8

[3] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8126–8135, 2021. 2

[4] Yongbiao Chen, Sheng Zhang, Fangxin Liu, Zhigang Chang, Mang Ye, and Zhengwei Qi. Transhash: Transformer-based hamming hashing for efficient image retrieval. *arXiv preprint arXiv:2105.01823*, 2021. 2, 4, 5, 7, 8

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 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*, pages 1–9, 2009. 5

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 2, 3, 4

[7] Shiv Ram Dubey. A decade survey of content based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021. 1

[8] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021. 2

[9] Lixin Fan, KamWoh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised

learning of accurate binary hashing codes. In *IJCAI*, pages 825–831, 2020. 1, 4, 5, 7, 8

[10] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1

[11] Socratis Gkelios, Yiannis Boutalis, and Savvas A Chatzichristofis. Investigating the vision transformer model for image retrieval tasks. *arXiv preprint arXiv:2101.03771*, 2021. 2

[12] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020. 2

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 4

[15] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021. 2

[16] Bumsoo Kim, Junhyun Lee, Jaewoo Kang, Eun-Sol Kim, and Hyunwoo J Kim. Hotr: End-to-end human-object interaction detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 74–83, 2021. 2

[17] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 4

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1, 4

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1

[20] Shuyan Li, Xiu Li, Jiwen Lu, and Jie Zhou. Self-supervised video hashing via bidirectional transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13549–13558, 2021. 2

[21] Shaohua Li, Xiuchao Sui, Xiangde Luo, Xinxing Xu, Yong Liu, and Rick Siow Mong Goh. Medical image segmentation using squeeze-and-expansion transformers. *arXiv preprint arXiv:2105.09511*, 2021. 2

[22] Yulin Li, Jianfeng He, Tianzhu Zhang, Xiang Liu, Yongdong Zhang, and Feng Wu. Diverse part discovery: Occluded person re-identification with part-aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2898–2907, 2021. 2

[23] Y Li, W Pei, Yufei Zha, and JC van Gemert. Push for quantization: Deep fisher hashing. In *30th British Machine Vision Conference, BMVC 2019*, 2020. 7, 8

[24] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021. 2

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[26] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2064–2072, 2016. 1, 4, 5, 7, 8

[27] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018. 1

[28] Xiaofeng Liu, Xiongchang Liu, Bo Hu, Wenxuan Ji, Fangxu Xing, Jun Lu, Jane You, C-C Jay Kuo, Georges El Fakhri, and Jonghye Woo. Subtype-aware unsupervised domain adaptation for medical diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2189–2197, 2021. 1

[29] Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. Thinking fast and slow: Efficient text-to-visual retrieval with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9826–9836, 2021. 2

[30] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010. 1

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *arXiv preprint arXiv:2103.13413*, 2021. 2

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the Advances in neural information processing systems*, pages 91–99, 2015. 1

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4

[34] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 806–815, 2018. 1, 4, 5, 7, 8

[35] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014. 1

[36] Fuwen Tan, Jiangbo Yuan, and Vicente Ordonez. Instance-level image retrieval using reranking transformers. *arXiv preprint arXiv:2103.12236*, 2021. 2

[37] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[39] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1571–1580, 2021. 2

[40] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3083–3092, 2020. 1, 4, 5, 7, 8

[41] Zheng Zhang, Qin Zou, Yuewei Lin, Long Chen, and Song Wang. Improved deep hashing with soft pairwise similarity for multi-label image retrieval. *IEEE Transactions on Multimedia*, 22(2):540–553, 2019. 1, 4, 5

[42] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. 4, 7, 8