1-2024

Efficient unsupervised video hashing with contextual modeling and structural controlling

Jingru DUAN

Yanbin HAO

Bin ZHU
Singapore Management University, binzhu@smu.edu.sg

Lechao CHENG

Pengyuan ZHOU

See next page for additional authors

Efficient Unsupervised Video Hashing with Contextual Modeling and Structural Controlling

Jingru Duan, Yanbin Hao, Bin Zhu, Lechao Cheng, Pengyuan Zhou, Xiang Wang

***Abstract*—The most important effect of the video hashing technique is to support fast retrieval, which is benefiting from the high efficiency of binary calculation. Current video hash approaches are thus mainly targeted at learning compact binary codes to represent video content accurately. However, they may overlook the generation efficiency for hash codes, i.e., designing lightweight neural networks. This paper proposes an *Efficient Unsupervised Video Hashing (EUVH)* method, which is not only for computing compact hash codes but also for designing a lightweight deep model. Specifically, we present an MLP-based model, where the video tensor is split into several groups and multiple axial contexts are explored to separately refine them in parallel. The axial contexts are referred to as the dynamics aggregated from different axial scales, including long/middle/short-range dependencies. The group operation significantly reduces the computational cost of the MLP backbone. Moreover, to achieve compact video hash codes, three structural losses are utilized. As demonstrated by the experiment, the three structures are highly complementary for approximating the real data structure. We conduct extensive experiments on three benchmark datasets for the unsupervised video hashing task and show the superior trade-off between performance and computational cost of our EUVH to the state of the arts.**

***Index Terms*—Video hashing, Deep Neural Network, Data Structure, Large-scale retrieval.**

I. Introduction

With the development of information technology, the Internet is filled with a large amount of video data. There's an increasing demand for effective and efficient video retrieval techniques. Hashing technology, which has the characteristics of low memory cost and fast retrieval speed, shows great potential for real-time content-based retrieval [1]–[7]. This paper studies unsupervised video hashing and focuses on fast video retrieval. Compared to the supervised scenario, unsupervised video hashing eliminates the necessity for manual annotations. It is thus challenging as approaches generally require approximating underlying video similarities for content-based retrieval [8], [9].

Video hashing aims to represent video content with a set of binary codes and achieve fast retrieval using Hamming distance. The hash codes are thus expected to embody the informative content of a video. It gives the first challenge of capturing video content as much as possible. This is also the main concern of existing methods. To meet this challenge, previous video hashing approaches make diligent efforts to extract rich visual features and build various sophisticated mathematical models. For example, MFH [10] and SMVH [11] combine the global color feature HSV [12] and local texture pattern LBP [13] for near-duplicate video retrieval. The more recent BTH [8] proposes to use the advanced bidirectional transformer network to well capture the long-range visual dependencies from both forward and backward directions. Generally, these works concern the high retrieval accuracy most. They have largely ignored the need for an efficient hash code generation model.

Another challenge is the data structure approximation. This is essential for a content-based retrieval task. Unsupervised video hashing requires to be able to approximate the inherent video relevance structure and reproduce it based on hash codes. Data structures over a video dataset can vary from global to local similarity patterns. In earlier studies, hashing approaches, e.g., SMVH [14], self-supervised video hashing (SSVH) [15], and neighborhood preserving hashing (NPH) [16], mainly focus on capturing local neighborhood relationships. It is usually beneficial to top-k retrieval precision, especially for a small k, e.g., k=1,2,5,10. However, the distribution of the holistic samples could not be well captured within a small group of neighbors when considering local neighborhood only [11], [17]. Late on, hashing methods [8], [9] takes the cluster as a complementary to the local neighborhood, which captures statistics reflecting cohort characteristics of whole samples. Actually, data structure modeling is still an open question for unsupervised video hashing methods.

In this paper, we propose an efficient unsupervised video hashing (EUVH) method, where efficiency here refers to the lightweight model. Specifically, we adopt the MLP-Mixer [18] as the model backbone and equip it with the group contextualization (GC) [19], named GC-MLP. In fact, GC-MLP shares some spirit with multi-granularity contextualized MLP (MC-MLP) [9] but possesses its uniqueness and can achieve higher model efficiency and retrieval accuracy. The model efficiency is achieved by using the channel splitting mechanism. As demonstrated by [19], the group contextualization decomposes the feature channels into several groups and separately uses different feature calibration operations to refine plain video features in parallel. It not only results in a lightweight neural network model but also enables the efficient use of multi-granularity contexts. Particularly, three kinds of axial contexts

Manuscript received February 20, 2023; revised November 5, 2023; accepted February 11, 2024.

J. Duan and X. Wang are with School of Cyber Science and Technology, University of Science and Technology of China, Hefei, China (e-mail: duanjr@mail.ustc.edu.cn; xiangwang1223@gmail.com).

Y. Hao and P. Zhou are with School of Information Science and Technology, University of Science and Technology of China, Hefei, China (e-mail: haoyanbin@hotmail.com; zpymyyn@gmail.com).

B. Zhu is with School of Computing and Information Systems, Singapore Management University, Singapore (e-mail: binzhu@smu.edu.sg).

L. Cheng is with School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China (e-mail: chenglc@hfut.edu.cn).

are exploited for EUVH, including long/middle/short-range dependencies. These video content dependencies are referred to as dynamic statistics aggregated over different time scales and can probably acclimatize various video activities.

EUVH addresses the second challenge by introducing three distinct data structures, including cluster, local neighborhood similarity, and contrast relation. The learning pipeline of hashing is to approximate these structures in the original video feature space and reconstruct them in the hash code space. We call it structural controlling. The three structures consider different structural information. Particularly, cluster shows the global cohort characteristics of whole samples, local neighborhood similarity reflects the local relationship within a small group of neighbors [20]–[22], while contrast relation emphasizes the relationship comparison between positive and negative examples [23]. As demonstrated by our experimental results, the combination of three structures leads to more discriminative hash codes, which also verify their complementarities in modeling real data distribution.

We summarize the contributions of our paper below:

- A lightweight MLP-based video hashing neural network GC-MLP is proposed, which explicitly exploits three kinds of axial contexts and separately refines video feature groups in parallel. The channel grouping operation significantly reduces the computation cost.
- Three distinct structures are constructed to reflect the real video data distributions, covering the global cohort characteristics, the local neighborhood similarity, and the contrast information. Accordingly, three loss functions are designed to control the structure transformation between the original feature space to the hash code space.
- Extensive experiments are conducted on three benchmarked video datasets for content-based video retrieval. The results show that our EUVH not only achieves better or more competitive performances but also significantly reduces the computation cost (number of parameters and FLOPs).

II. Related Work

In this section, we briefly review some existing hashing works, most of which are adopted for data representation learning. We refer to these works as learning-based hashing in contrast to data-independent hashing. Since our hashing method targets video content processing, we thus give a more detailed review for video hashing.

A. learning-based hashing

The learning-based hashing generally designs a data-dependent framework to learn binary hash code representation. According to the learning fashion, it can be classified into two categories: supervised hashing and unsupervised hashing.

Supervised hashing learns semantically discriminative hash codes by using manual annotations. CCA-ITQ [24] uses the Canonical Correlation Analysis (CCA) to maximize the correlation between semantically similar features and utilizes ITQ to minimize the quantization loss. Kernel-based supervised hashing (KSH) [25] performs a kernel function to map the data to compact binary codes by minimizing the distance of similar pairs and maximizing the distance of dissimilar pairs. Supervised discrete hashing (SDH) [26] proposes a discrete cyclic coordinate descent algorithm to solve the discrete optimization without any relaxations. Deep supervised hashing (DSH) [27] learns hash codes using pairs of samples and proposes a regularizer to reduce the disparity between real-valued vectors and discrete binary vectors. Deep asymmetric pairwise hashing (DAPH) [28] and Asymmetric deep supervised hashing (ADSH) [29] use the asymmetric hashing approach to learn the semantic information rich hash functions.

Unsupervised hashing mainly discovers the relationship between data by exploring data structure. Spectral hashing (SH) [30] and self-taught hashing (STH) [31] learn hash functions by binarizing the eigenvectors of the graph. PCA-ITQ [24] utilizes Principal Component Analysis (PCA) to reduce the dimension and minimizes the quantization loss by a rotation matrix. K-means hashing (KMH) [32] proposes a novel quantization algorithm to encode and quantize, considering both quantization and distance approximation. Anchor graph hashing (AGH) [33] uses anchor graphs to detect the data's inherent neighborhood similarity and learn compact hash codes. To learn hash codes, deep hashing (DH) [34] develops a deep neural network that seeks multiple hierarchical non-linear transformations. Similarity-adaptive discrete hashing (SADH) [35] proposes a similarity graph updating phase to preserve the data similarities and solves the non-smooth discrete hashing problem with the efficient alternating direction method of the multipliers algorithm. Deep ordinal hashing (DOH) [7] explores the ranking structure from both local and global views to generate ranking-based hash codes for images. The work [6] proposes instance-aware image representations for achieving both semantic and category-aware hashing.

B. video hashing

Video hashing is to compress high-dimensional and complex video content into simple binary codes. Multiple feature hashing (MFH) [10] is based on spectral factorization for model optimization and learns hash function by weighting multiple features of different types. Submodular video hashing (SVH) [36] learns hash function based on a selection of relevant frames from the video. Video hashing with both discriminative commonality and temporal consistency (VHDT) [37] takes into account the temporal consistency when learning the hash codes. Stochastic multiview hashing (SMVH) [14] and its extension t-distributed SMVH (t-SMVH) [11] use Kullback-Leibler (KL) divergence to achieve similar structure preservation from feature space to Hamming space (i.e., the hash code space).

In recent years, thanks to the development of powerful computation machines, deep learning techniques, e.g., RNNs, CNNs [38], MLPs [18], and transformers [39], have been applied in various fields and achieved improved achievements. Deep video hashing (DVH) [40] learns the hash function by maximizing the inter-class distance and minimizing the intra-class distance of video pairs. Similarity-preserving deep temporal hashing (SPDTH) [41] utilizes an end-to-end supervised

video retrieval framework to learn discriminative hash codes and solves the slow convergence problem by enabling positive pairs to be compared with all corresponding negative pairs. Self-supervised temporal hashing (SSTH) [42] constructs an unsupervised deep video hashing model and reconstructs the original frame features from the relaxed hash codes. Self-supervised video hashing (SSVH) [15] improves SSTH by building a neighborhood structure to capture the neighborhood similarity information between videos. Joint appearance and temporal encoding (JTAE) [43] combines two kinds of information to facilitate video hashing by collectively learning two encoders. [44] takes CNN features and then uses deep learning to learn hash function, while [45] explores video information by additionally extracting optical flow features, which prove to be useful in video understanding. Unsupervised variational video hashing (UVVH) [46] selects CNN encoder to generate hash codes and reconstructs the frame-level features from hash codes by CNN-LSTM decoder, and uses a probabilistic latent loss to approximate the posterior distribution to a prior. Neighborhood preserving hashing (NPH) [16] proposes a neighborhood attention mechanism to promote hash codes to capture the neighborhood-relevant content under the guidance of each input video's spatio-temporal neighborhood information. structure-adaptive neighborhood preserving hashing (SNPH) [17] extends NPH by using two-layer LSTMs to leverage the hierarchical structure of the video and takes cluster assignment into account to learn more discriminative hash codes. Hashing method based on bidirectional transformers (BTH) [8] utilizes the bidirectional transformers to exploit the long-range bidirectional correlations between frames and introduces a new similarity to distinguish boundary samples. Multi-Structure preserved Hashing (MCMSH) [9] designs three self-gating modules to explore multiple axial contexts and integrates them into MLP-Mixer for feature contextualization and adopts three types of data structures to investigate video content for discriminative video hash codes. Dual-stream knowledge preserving hashing (DKPH) [47] constructs a dual-stream structure to investigate video information for the semantic dependent binary codes generation, and proposes a Gaussian-adaptive similarity graph to preserve semantic similarity in the Hamming space.

III. Proposed Method

In this section, we elaborate on the proposed Efficient Unsupervised Video Hashing (EUVH) method. An overview of the proposed EUVH is illustrated in Figure 1. First, we introduce the structure of the proposed model. The structural exploration of video data is then presented. Finally, we provide the formula for model optimization.

A. Contextual Modeling

Given N videos $\mathcal{V} = \{\mathbf{V}_i\}_{i=1}^{N}$, we sample T frames from each video and process each frame with VGG [22] to generate CNN features. The i-th video is thus represented by a feature matrix $\mathbf{V}_i = \left[\boldsymbol{v}_1^i, \boldsymbol{v}_2^i, \cdots, \boldsymbol{v}_T^i\right]^T \in \mathbb{R}^{T\times C}$, where C denotes the channel dimension corresponding to a frame. The extracted feature matrix $\mathbf{V}_i$ is regarded as the input of EUVH. Our goal is to learn a binary code $\boldsymbol{b}_i \in \{-1, 1\}^K$ for each video input, where K is the length of the code.

*1) **Overview of GC-MLP**:* GC-MLP adopts the MLP-Mixer [18] as its model backbone and is equipped with group contextualization [19], which achieves higher retrieval accuracy and model efficiency. MLP-Mixer takes "$patches \times channels$" as input and employs a channel-mixing MLP layer and a token-mixing MLP layer to interact between distinct channels and tokens successively, achieving equivalent performance to CNNs and transformers but at a reduced computational cost. Although MLP-Mixer can communicate between channels and between tokens, we believe it will benefit feature learning when further investigating different axial contexts. Specifically, MC-MLP proposes to utilize three kinds of self-gating modules, a long-range dependency (L-RD) module, a middle-range dependency (M-RD) module, and a short-range dependency (S-RD) module, to model achieve video feature contextualization. The three dependency modules focus on the use of statistical dynamics. To improve model efficiency, we adopt the grouping strategy to achieve parallel feature contextualization. We present the three dependency modeling modules and the grouping strategy in detail below. It should be noted that due to the high channel dimension C (4,096 here) of the input CNN feature, we utilize a fully connected (FC) layer followed by the activation function ReLU to reduce C to a smaller number D (e.g., 256) before feeding the feature to the model.

*2) **Group Contextualization**:* The group contextualization [19] divides the feature channels into several groups and separately refines them in parallel using different axial dependencies. Unlike MC-MLP, which utilizes each self-gating module to handle the entire features and then merges the results, GC-MLP employs the three self-gating modules L/M/S-RD to separately process feature groups in parallel and then concatenates the results, resulting in a less computationally intensive model. In the implementation, we split the feature channels into four groups, then utilize the L/M/S-RD modules to separately process the first three groups in parallel, keep the fourth group unprocessed, and finally connect them together to feed into the next layer of the neural network.

The detailed architectures of L/M/S-RD modules are depicted in Figure 2. They share a similar bottleneck structure of "Pooling/None $\rightarrow$ FC/Convolution+ReLU $\rightarrow$ FC/Convolution+Sigmoid" that achieves different scales of context modeling with pooling operations. Specifically, the L-RD module obtains the global axial context by processing the feature group through a global 1D average pooling layer along the time/token axis. Then it uses the FC layer to achieve dimensional reduction and channel/token interaction. The M-RD module handles the feature group via an average pooling layer with a smaller kernel to retrieve the middle-range content. Different from L-RD, M-RD processes the resulting tensor using two 1D convolutions with a 3 kernel instead of the FC layer based on the tensor size. Finally, unlike L-RD and M-RD, S-RD focuses on local information without using average pooling operation. It directly uses a 1D convolution with a 3 kernel to capture context from 3 neighborhood tokens. Note that we use r to reduce the feature dimension in the first

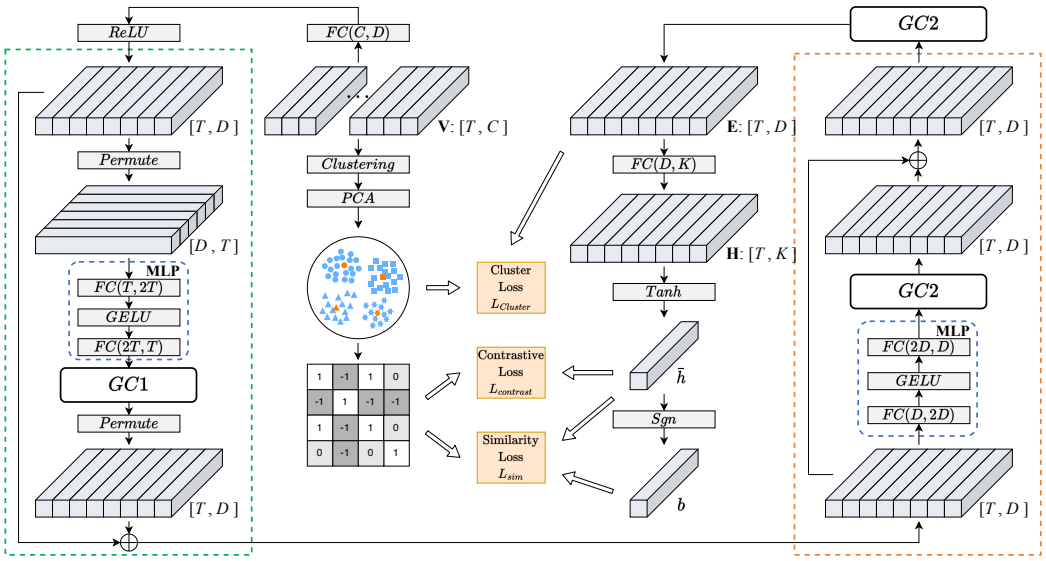


Fig. 1. Overall structure of the proposed EUVH. The orange dashed box represents the channel-mixing FC layer, and the green dashed box represents the token-mixing FC layer. The input is a feature matrix $\mathbf{V}$, and the output is a binary code vector $\boldsymbol{b}$.

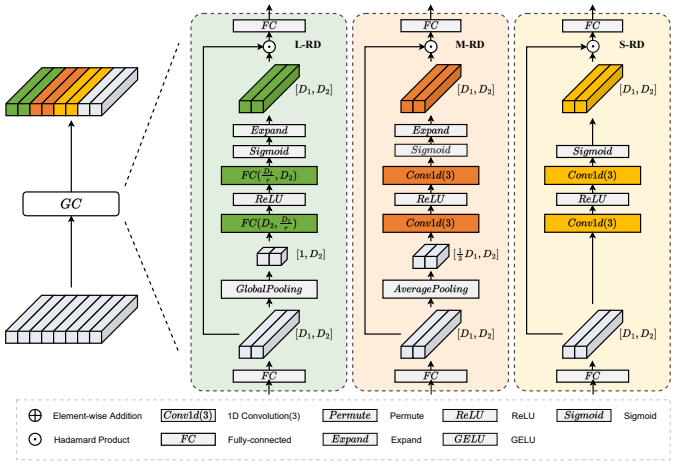


Fig. 2. The structure of the Group Contextualization.

convolution or FC layer.

In addition, we add an FC layer before and after the gating unit, respectively. It should be noted that MLP-Mixer has two kinds of layers: the channel-mixing FC layer (i.e., the orange dashed box in Figure 1) and the token-mixing FC layer (i.e., the green dashed box in Figure 1). In the token-mixing FC layer, "$patches \times channels$" will be transposed to "$channels \times patches$", but the above formula is still applicable, only the reduction ratio r will be applied with a different value. Particularly, the feature group $D_1 \times D_2$ is $T \times \frac{D}{4}$ in the channel-mixing FC layer and $\frac{D}{4} \times T$ in the token-mixing FC layer. In practice, the first self-gating modules share a similar spirit with the squeeze-and-excitation network (SE-Net) [48], where the axial contexts are dynamics aggregated statically within a length-fixed axial duration.

*3) **Hash layer**:* We can obtain the latent video feature matrix $\mathbf{E}_i = \left[\boldsymbol{e}_1^i, \boldsymbol{e}_2^i, \cdots, \boldsymbol{e}_T^i\right]^T \in \mathbb{R}^{T \times D}$ by feeding the input video feature matrix $\mathbf{V}_i$ into the proposed GC-MLP. Next, we design a hash layer to reduce the dimension and discretize the latent embedding matrix $\mathbf{E}_i$. We firstly use a FC layer to project $\mathbf{E}_i$ to a real-valued feature matrix $\mathbf{H}_i = \left[\boldsymbol{h}_1^i, \boldsymbol{h}_2^i, \cdots, \boldsymbol{h}_T^i\right]^T \in \mathbb{R}^{T \times K}$, where K denotes the length of hash code. Then, we fuse $\mathbf{H}_i$ through mean pooling, and map it to the range of $(-1, 1)$ through the Tanh function to get a relaxed binary vector $\bar{\boldsymbol{h}}_i$. Finally, we utilize Sgn to discretize $\bar{\boldsymbol{h}}_i$ to a binary vector $\boldsymbol{b}_i$. The calculation is as follows

$$\mathbf{H}_i = \mathrm{FC}(\mathbf{E}_i), \tag{1}$$

$$\bar{\boldsymbol{h}}_i = \mathrm{Tanh}(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{h}_t^i), \tag{2}$$

$$\boldsymbol{b}_i = \mathrm{Sgn}(\bar{\boldsymbol{h}}_i), \tag{3}$$

where $\mathrm{Sgn}(x) = 1$ if $x \geq 0$ and $\mathrm{Sgn}(x) = -1$ otherwise. For activation function Tanh, we follow [49]–[51] to use the scaled Tanh function as follows

$$\bar{\boldsymbol{h}}_i = \mathrm{Tanh}(\rho\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{h}_t^i), \tag{4}$$

where the ρ keeps increasing during training. In mathematics, there is a relationship between the Sgn function and the scaled Tanh function: $\lim_{\rho\to\infty}\mathrm{Tanh}(\rho z) = \mathrm{Sgn}(z)$, where $\rho > 0$. Thus, as $\rho \to \infty$, the scaled Tanh function addresses the optimization problem of the Sgn function very well.

B. Structural Controlling

To capture the real video data distributions and learn more discriminative hash codes, we construct three distinct data structures, including cluster, local neighborhood similarity, and contrast relation.

*1) **Cluster**:* Clustering is the process of dividing a data set into several classes based on a certain standard (e.g., distance) in order to maximize the similarity of data items in the same class and minimize the difference of data objects not in the same class. Therefore, the clustering algorithm can naturally form a pseudo-label for unlabeled data and cluster structure displays the global cohort characteristics of entire samples. In the implementation, we first apply average pooling to the input video feature matrix $\mathbf{V}_i$ and the latent feature matrix $\mathbf{E}_i$ to produce the video vector embedding $\bar{\boldsymbol{v}}_i = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{v}_t^i$ and the latent video representation $\bar{\boldsymbol{e}}_i = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{e}_t^i$ respectively. After that, we use K-means to cluster N video vector embeddings to generate M cluster centers $\{\boldsymbol{u}_j\}_{j=1}^{M}$, and then apply PCA to decrease dimension C of cluster centers $\{\boldsymbol{u}_j\}_{j=1}^{M}$ to the same dimension D as $\bar{\boldsymbol{e}}_i$. Finally, we align the learnt video representation $\bar{\boldsymbol{e}}_i$ with its the nearest class center $\boldsymbol{u}_i^1$, and calculate the cluster loss $\mathcal{L}_{cluster}$ as shown below

$$\mathcal{L}_{cluster} = \frac{1}{B}\sum_{i=1}^{B} \|\bar{\boldsymbol{e}}_i - \boldsymbol{u}_i^1\|_2^2, \tag{5}$$

where B is the training batch size.

*2) **Local neighborhood similarity**:* The relationship between a sample and its class center is the primary emphasis of cluster structure, which does not consider the correlation between data pairs. The goal of local neighborhood similarity structure is thus to construct a pairwise similarity graph $\mathbf{S} \in \mathbb{R}^{N\times N}$ with each element reflecting the similarity of two samples.

We follow the works [8], [33] to build a similarity graph $\mathbf{S}$. Specifically, we first calculate m nearest cluster centers $\{\boldsymbol{u}_i^1, \boldsymbol{u}_i^2, \cdots, \boldsymbol{u}_i^m\}$ for each video vector embedding $\bar{\boldsymbol{v}}_i$ and calculate a truncated similarity matrix $\mathbf{P} \in \mathbb{R}^{N\times M}$ by the following formula

$$\mathbf{P}_{i,j} = \begin{cases} \dfrac{\exp(-\|\bar{\boldsymbol{v}}_i - \boldsymbol{u}_i^j\|_2^2/\sigma)}{\sum_{l=1}^{m}\exp(-\|\bar{\boldsymbol{v}}_i - \boldsymbol{u}_i^l\|_2^2/\sigma)}, & \forall \boldsymbol{u}_i^j \in \{\boldsymbol{u}_i^*\}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $\|\cdot\|_2$ denotes the l_2-norm, and σ is a bandwidth parameter. Then, we calculate a adjacency matrix $\mathbf{A} = \mathbf{P}\Lambda^{-1}\mathbf{P}^T \in \mathbb{R}^{N\times N}$, where $\Lambda = \text{diag}(\mathbf{P}^T\mathbf{1}) \in \mathbb{R}^{M\times M}$. And we construct a binary similarity matrix $\mathbf{A}'$ according to $\mathbf{A}'_{ij} = 1$ if $\mathbf{A}_{ij} > 0$ and $\mathbf{A}'_{ij} = -1$ otherwise, and the binary similarity matrix $\mathbf{A}'$ are determined by the number of nearest cluster centers. In practice, we follow [8] and set three numbers m_1, m_2, m_3 ($m_1 < m_2 < m_3$) for m, and obtain three corresponding matrices $\mathbf{A}'^{(1)}$, $\mathbf{A}'^{(2)}$ and $\mathbf{A}'^{(3)}$. Finally, we construct the ultimate similarity matrix $\mathbf{S}$ as

$$\mathbf{S}_{ij} = \begin{cases} 1, & \mathbf{A}'^{(1)} = 1, \\ -1, & \mathbf{A}'^{(2)} = -1 \text{ and } \mathbf{A}'^{(3)} = 1, \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

For local neighborhood similarity structure, positive and negative samples are easily confused and misjudged at the boundary. So obtaining reliable pairwise similarity at the boundary is more advantageous for learning discriminative hash codes. In addition, due to a large number of negative samples, compared with the binary similarity matrix $\mathbf{A}'$, the ultimate similarity matrix $\mathbf{S}$ adds a new similarity degree of 0 to differentiate boundary cases and also reduce the possible bias caused by too many negative samples.

Since the Sgn function makes the optimization intractable, we compute the cosine similarity between the relaxed binary vectors $\bar{\boldsymbol{h}}_i$ and $\bar{\boldsymbol{h}}_j$ as $\frac{1}{K}\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_j$ and involve a quantization error as a penalty term to narrow the gap between the relaxed binary vectors and the binary vectors. The local neighborhood similarity loss is shown as follows

$$\mathcal{L}_{sim} = \frac{1}{B}\sum_{\mathbf{S}_{ij}\in\mathbf{S}} (\frac{1}{K}\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_j - \mathbf{S}_{ij})^2 + \varepsilon\frac{1}{B}\sum_{i=1}^{B}\|\boldsymbol{b}_i - \bar{\boldsymbol{h}}_i\|_2^2. \tag{8}$$

The coefficient ε is set to 0.1 following [9]. Since the hash code vector $\boldsymbol{b}_i$ is computed by Sgn which is non-differentiable, we use straight-through estimator (STE) [52] to deal with the problem of non-differentiable during back-propagation.

*3) **Contrast relation**:* People frequently learn to discriminate between objects by comparing them. Similarly, contrastive learning requires the model to learn high-level features that are sufficient to differentiate objects rather than precise details. The purpose of contrast relation structure is to bring the augmented data of the same sample close to each other while pushing different samples away in the embedding space. Due to the great success of [23], we leverage contrast relation for structural control to allow the model to learn more discriminative hash codes. The difficulty of contrastive learning is how to construct positive and negative samples. In the field of images, there are some general methods such as rotation and cropping, and in the field of text, frequently used methods such as back-translation, character insertion or deletion. Because each video comprises several frames, positive samples can be created by extracting varying quantities of frames. The real-valued vector $\{\boldsymbol{h}_t^i\}_{t=1}^{T}$ contains T frames feature vector of a video, in the implementation, we randomly extract T' frames from the T frames and then fuse them through average pooling. Finally, we can obtain the augmented sample $\bar{\boldsymbol{h}}'_i$ corresponding to the relaxed binary vector $\bar{\boldsymbol{h}}_i$. Using the data augmentation strategy described above, we can get $2B$ data points in a batch of B samples. Similar to [23], we have a positive pair and treat the other $2(B-1)$ as negative samples in a batch. We use $sim(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}'_i) = \frac{\bar{\boldsymbol{h}}_i\cdot\bar{\boldsymbol{h}}'_i}{\|\bar{\boldsymbol{h}}_i\|_2\|\bar{\boldsymbol{h}}'_i\|_2}$ to represent the cosine similarity of $\bar{\boldsymbol{h}}_i$ and $\bar{\boldsymbol{h}}'_i$, and follow InfoNCE [53] to design the contrastive loss as follows

$$\bar{\boldsymbol{h}}'_i = \text{Tanh}(\frac{1}{T'}\sum_{t=1}^{T'}\boldsymbol{h}_t'^{\boldsymbol{i}}), \tag{9}$$

$$\mathcal{L}_{contrast} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\mathfrak{E}(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}'_i)}{\sum_{k\neq i}\mathfrak{E}(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}_k) + \sum\mathfrak{E}(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}'_k)}, \tag{10}$$

where $\mathfrak{E}(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}'_i) = \exp(sim(\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{h}}'_i)/\tau)$ and the temperature parameter τ is set 0.5 following [23].

*4) **Discussion**:* The three data structures, i.e., cluster, local neighborhood similarity, and contrast relation, exhibit various structural controls in hash function learning. Specifically,

Algorithm 1: The training of EUVH

Input: Input video feature $\{\mathbf{V}_i\}_{i=1}^{N}$, epoch numbers $Epoch$, hyperparameters
Output: Network parameters of EUVH

```
// Initialization
for i = 1 to N do
    Obtain video embedding v̄_i by applying average pooling on input video feature V_i;
    Gain cluster centers {u_j}_{j=1}^{M} by employing K-means on video embedding v̄_i;
    Build similarity matrix S with Eq. 6 and Eq. 7;
    Initialize network parameters of EUVH;
// Training
for epoch = 1 to Epoch do
    for i = 1 to N do
        Generate latent video feature E_i = GC-MLP(V_i);
        Compute real-valued feature H_i by Eq.1;
        Calculate relaxed binary vector h̄_i with Eq.2;
        Obtain hash code b_i by Eq.3;
        Gain ē_i by applying average pooling on latent video feature E_i;
        Calculate h̄'_i with Eq. 9;
        Update network parameters by minimizing the overall loss (Eq.11);
```

local neighborhood similarity structure takes into account the pairwise similarity within a small region, reflecting a local neighborhood relationship. However, when only local neighborhood similarity structure is considered, the distribution of the entire sample cannot be fully captured. As complementary to local neighborhood similarity, cluster captures the global cohort characteristics of total samples, encouraging a sample to congregate its allocated cluster centers. Thus, the cluster and local neighborhood similarity are primarily concerned with the relationship between samples. While the contrast relation structure focuses on the compared information between sample pairs, that is, the similarity of the anchor sample to its positive target should be larger than that to the negative target. Besides, our experimental results show that the combination of three structures leads to superior retrieval performance and also demonstrate the complementarity nature of the three structures.

C. Model optimization

GC-MLP models the input video features by exploiting three different types of axial contexts to generate latent video features, and then three complementary structures are employed to approximate the inherent video relevance structure and reflect the real video data distributions. The training of EUVH is illustrated in Algorithm 1. The model optimization is based on a multi-objective formulation that combines the three different loss functions, and the overall loss for optimization is as follows

$$\mathcal{L}_{all} = \alpha\mathcal{L}_{cluster} + \beta\mathcal{L}_{sim} + \gamma\mathcal{L}_{contrast}, \tag{11}$$

where α, β, and γ are hyper-parameters that balance above-mentioned three loss functions. We perform backpropagation on the above formula to optimize our model.

IV. Experiment

We conduct extensive experiments to verify the effectiveness and efficiency of our proposed method, including comparison with MCMSH, ablation study, and the comparison with the state of the arts, as well as some extension experiments.

A. Datasets

- **FCVID**. Fudan-Columbia Video Dataset [54] includes 91,223 YouTube videos manually annotated into 239 categories. The videos in this dataset depict a variety of activities, scenes, events, etc. We follow [15] to use 91,185 videos, with 45,585 videos used for training and the remaining 45,600 videos used for evaluation.
- **ActivityNet**. ActivityNet [55] offers 28k videos of 203 activity categories collected from YouTube. We follow [16] to use 9,722 videos for training and 4,758 videos for evaluation, with 1,000 videos serving as queries and the rest 3,758 videos serving as retrieval databases.
- **YFCC**. Yahoo Flickr Creative Commons 100 Million Dataset [56] consists of 0.8M videos divided into 80 categories from Flickr. We follow [42] to use 101,256 videos, where 409,788 videos are for training and 101,256 videos are for evaluation.

B. Evaluation Metrics

We follow [42] to employ Average Precision at top-k retrieval videos (AP@k) to evaluate the retrieval performance [57]. AP@k is defined as $\frac{1}{min(R,\mathrm{k})}\sum_{i=1}^{\mathrm{k}}\frac{R_i}{i}\times I_i$, where R denotes the number of relevant videos in the database, R_i represents the number of relevant videos in the top-i videos, and $I_i = 1$ if the i-th video is relevant and $I_i = 0$ otherwise. Then, we use the mean of AP@k over all queries (mAP@k) as the final evaluation metric.

C. Implementation Details

In the preprocessing stage, to be consistent with the setting of previous work, we sample $T = 25$ frames for each video in FCVID and YFCC datasets and utilize VGG-16 [22] network pre-trained on ImageNet [58] to extract 4096-D ($C = 4096$) frame features. For ActivityNet dataset, we follow [43] to sample $T = 30$ frames per video and use ResNet-50 [38] to generate 2048-D ($C = 2048$) frame features. The FCVID and YFCC datasets features are provided directly by [15] and the features of the ActivityNet dataset are provided directly by [8]. For model architecture, we set the reduced feature dimension to $D = 256$, and set the reduction ratio r used in L/M/S-RD modules as 8 for the channel-mixing layer and 4 for the token-mixing layer. For loss functions, we set the number of clusters as 2,000 and the number of nearest cluster centers $m_1 = 3, m_2 = 4, m_3 = 5$ following [8], and the randomly extracted number of frames $T' = 20$ from $T = 25$ or $T = 30$ frames. The balancing parameters α, β, γ used in

Eq. (11) are set as $\alpha = 0.8, \beta = 0.1, \gamma = 0.1$. During the training stage, we set batch size as 256 for FCVID and YFCC datasets and 64 for ActivityNet datasets, epochs as 60, the learning rate as 0.0003, and use Adam [59] algorithm with momentum 0.9 to update parameters. The code is available at https://github.com/duanjr666/EUVH.

D. Comparison with MCMSH

In this section, we compare our EUVH with the most relative method MCMSH in terms of retrieval accuracy and model complexity using FCVID dataset with hash code length of 64 bits.

TABLE I
PERFORMANCE (MAP@K) COMPARISON WITH MCMSH ON FCVID DATASET WITH 64 BITS CODES.

Model	k=5	k=20	k=40	k=60	k=80	k=100
MCMSH	0.494	0.335	0.288	0.263	0.245	0.230
MCMSH (L_{all})	**0.506**	0.349	0.302	0.277	0.259	0.243
EUVH	**0.506**	**0.351**	**0.305**	**0.280**	**0.261**	**0.245**

TABLE II
MODEL COMPLEXITY COMPARISON ON FCVID DATASET WITH 64 BITS CODES.

Model	Param.	FLOPs	Average Encoding Time
BTH	3.17M	0.05G	0.53ms
MCMSH	1.76M	0.05G	0.47ms
EUVH	1.37M	0.04G	0.20ms

Table I illustrates the performance of EUVH, the original MCMSH, and the modified MCMSH under the overall loss function Eq.(11), with the balancing parameters settings to $\alpha = 0.8$, $\beta = 0.1$ and $\gamma = 0.1$. We observe that EUVH and the modified MCMSH have similar performances (EUVH performs relatively better than MCMSH for k > 5). Compared with the original MCMSH, both EUVH and the modified MCMSH can significantly improve performance. Since we replace the triplet loss used by the original MCMSH with the contrastive loss for EUVH and the modified MCMSH, it thus demonstrates the superiority of modeling contrast relation.

Table II compares EUVH and the two current the-state-of-the-art methods BTH and MCMSH in terms of the numbers of parameters and FLOPs, and the average encoding time. We observe that EUVH has 22% fewer parameters than MCMSH and only 43% as much as BTH's. In terms of FLOPs, EUVH is also lower than BTH and MCMSH, indicating that EUVH is less computationally expensive than BTH and MCMSH. The average encoding time counts the time cost of converting a video feature to a hash code vector. It can be found that EUVH is much faster than BTH and MCMSH (0.20ms vs. 0.53ms of BTH and 0.47ms of MCMSH). By jointly taking into account the retrieval performance and computational costs, our EUVH achieves the best trade-off between performance and model complexity. This meets our intention of designing a more lightweight neural network model but without losing (actually improving) retrieval performance.

E. Ablation Study

In this section, we study the effect of hyperparameters, including the three self-gating modules L/M/S-RD, the dimension reduction ratio r used in the self-gating modules, the number of randomly extracted frames T' from T frames in one video and the balancing parameters α, β, γ used in Eq. (11), using the FCVID dataset.

Effect of different axial contexts. We first verify the effect of the three context learning modules (also known as self-gating modules) L/M/S-RD, which capture long-range, middle-range and short-range dependencies respectively. EUVH is compared to the following variants: (1) MLP-Mixer: the original MLP-Mixer architecture but with only one block (one channel-mixing layer and one token-mixing layer); (2) with L-RD. We use the MLP-Mixer as a backbone and only process the first feature group using the L-RD module; (3) with M-RD. Similar to (2), we employ the MLP-Mixer as a backbone and only handle the second feature group with the M-RD module; (4) with S-RD. As (2) and (3), the S-RD module is used to process the third feature group. The performances of these model variants are shown in Table III. Firstly, we observe that MLP-Mixer with a single context, e.g., L-RD, M-RD, or S-RD, can surpass the original MLP-Mixer, showing that calibrating video features with the axial context is useful for video content capturing. Based on these findings, we can conclude that the three self-gating modules can effectively model the data from various axis contexts. Then, when combining all three modules, EUVH obtains the best performance, indicating that the three self-gating modules can work together to improve performance.

Effect of dimension reduction ratio r. Secondly, we examine the effect of the dimension reduction ratio r used in the three self-gating modules. It should be noted that since there are only $T = 25$ frames in a video, we fix the setting of $r = 4$ following MCMSH for the token-mixing layer. So, we here only test different r settings for the channel-mixing layer, e.g., $r = 2, 4, 8, 16, 32$. Table IV lists the retrieval results and the number of parameters for the model variants with varying dimension reduction ratios r. We can see that the number of parameters decreases with r increases, but the performance fluctuates within a certain range. Considering the trade-off between the model size and the performance, we set the dimension reduction ratio r to 8 in the channel-mixing layer.

Effect of the number of randomly extracted frames T'. Thirdly, we test the effect of the number of randomly extracted frames T' from T frames used in Eq.10. The random selection strategy extracts T' ($T' < T$) frames randomly from T frames during training. Here we conduct experiments to test which T' is suitable for our model. Table V shows the performance of the various settings for T'. We can notice that when the number of randomly extracted frames T' increases, the performance has a trend of increasing first and decreasing then. As a result, we finally choose $T' = 20$ for our model.

Effect of different structures and their combinations. Finally, we explore the effect of the constructed data structures by varying the balancing parameters α, β, γ in Eq. (11). The

TABLE III
PERFORMANCE (MAP@K) COMPARISON WITH DIFFERENT EUVH VARIANTS ON FCVID WITH 32 BITS AND 64 BITS HASH CODES.

Model	32 bits					64 bits				
	k=20	k=40	k=60	k=80	k=100	k=20	k=40	k=60	k=80	k=100
MLP-Mixer	0.294	0.250	0.228	0.212	0.199	0.341	0.294	0.270	0.252	0.237
with L-RD	0.300	0.254	0.230	0.212	0.197	0.349	0.302	0.276	0.257	0.241
with M-RD	0.298	0.253	0.230	0.213	0.199	0.343	0.297	0.273	0.254	0.238
with S-RD	0.297	0.251	0.228	0.211	0.197	0.342	0.295	0.271	0.253	0.238
EUVH	**0.305**	**0.260**	**0.237**	**0.219**	**0.204**	**0.351**	**0.305**	**0.280**	**0.261**	**0.245**

TABLE IV
PERFORMANCE (MAP@K) COMPARISON OF EUVH WITH DIFFERENT DIMENSION REDUCTION RATIO r ON FCVID WITH 64 BITS CODE LENGTHS.

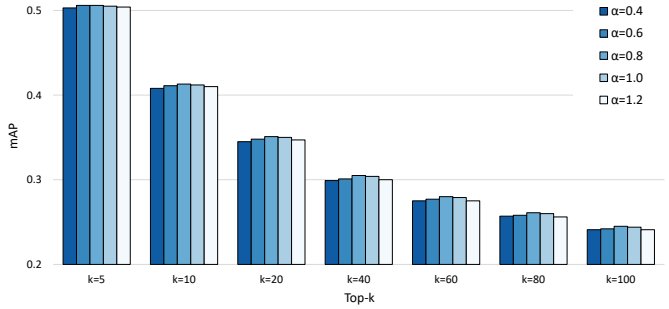
r	k=5	k=20	k=40	k=60	k=80	k=100	Param.
2	0.504	0.348	0.301	0.276	0.257	0.242	1.377M
4	0.501	0.346	0.300	0.275	0.257	0.241	1.372M
8	**0.506**	**0.351**	**0.305**	**0.280**	**0.261**	**0.245**	1.370M
16	0.504	0.347	0.300	0.276	0.257	0.242	1.369M
32	**0.506**	0.350	0.304	0.279	0.260	0.244	1.369M

TABLE V
PERFORMANCE (MAP@K) COMPARISON OF EUVH WITH THE RANDOMLY EXTRACTED DIFFERENT NUMBERS OF FRAMES T' ON FCVID WITH HASH CODE LENGTH 64 BITS.

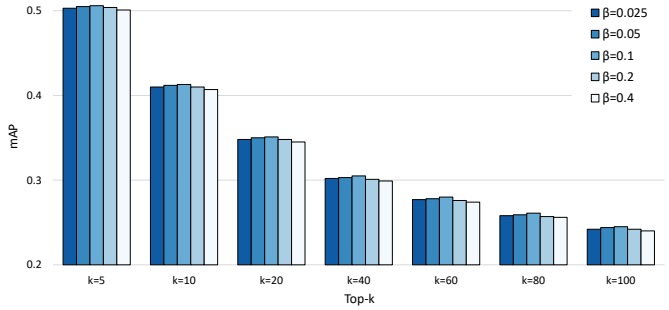
T'	k=5	k=20	k=40	k=60	k=80	k=100
5	0.505	0.348	0.301	0.276	0.257	0.241
10	0.505	0.348	0.301	0.276	0.257	0.242
15	0.505	0.348	0.301	0.276	0.258	0.242
20	**0.506**	**0.351**	**0.305**	**0.280**	**0.261**	**0.245**
25	**0.506**	0.350	0.303	0.278	0.260	0.244

three losses in Eq. (11) represent three kinds of structural information. They are the cluster controlled by α, local neighborhood similarity controlled by β, and contrast relation controlled by γ. Here, we first examine the performance of every single structure, and then their combinations. As shown in the first three rows of Table VI, the model with cluster loss only (i.e., $\alpha = 1$) achieves the best performance among the three structures. This is consistent with observations from BTH and MCMSH. One potential reason could be that the cluster provides the pseudo-label and thus makes the model training in a supervised fashion which can speed up the model convergence. Either the local neighborhood similarity or the contrast relation alone cannot result in satisfactory results. we secondly show the results of different structure pairs in the second three rows of Table VI. In general, all the structure pairs can significantly improve the performance compared with their corresponding single-structure counterparts. For example, when combining L_{cluser} and L_{sim}, the mAP@5 is improved from 0.458 (L_{cluser} only) and 0.416 (L_{sim} only) to 0.491. Another significant performance improvement (0.458/0.386→0.503) is also observed when combing L_{cluser} and $L_{contrast}$. Finally, when merging all three losses, EUVH obtains the best performance on all mAP@k (e.g., 0.506 on mAP@5). These performance trends show strong evidence that the three structures are complementary to each other and their combinations can greatly improve the performance.

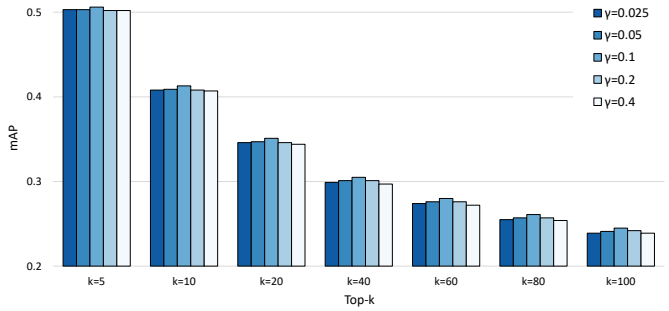
In addition, we also show the fine-grained tuning for the three hyperparameters $\{\alpha, \beta, \gamma\}$. Specifically, we set $\alpha = 0.4, 0.6, 0.8, 1.0, 1.2$, $\beta = 0.025, 0.05, 0.1, 0.2, 0.4$, and $\gamma = 0.025, 0.05, 0.1, 0.2, 0.4$, respectively. We test the performance change of one hyperparameter (e.g., α) by fixing the other two (e.g., β and γ). Figure 3 shows their corresponding performance changes. As observed, the setting of $\alpha = 0.8, \beta = 0.1, \gamma = 0.1$ can provide the best performance, which is also the ultimate setting of EUVH. In the following experiments, all results are obtained by using this ultimate setting.



(a) varying α with fixing β=0.1, γ=0.1



(b) varying β with fixing α=0.8, γ=0.1



(c) varying γ with fixing α=0.8, β=0.1

Fig. 3. Performance changes varying α (a), β (b) and γ (c) on FCVID with 64 bits hash codes.

TABLE VI
PERFORMANCE (MAP@K) COMPARISON WITH A SINGLE DATA STRUCTURE AND THEIR COMBINATION USING FCVID DATASET WITH 64 BITS CODES.

Loss	k=5	k=20	k=40	k=60	k=80	k=100
$L_{cluster}(\alpha=1)$	0.458	0.292	0.243	0.218	0.201	0.186
$L_{sim}(\beta=1)$	0.416	0.254	0.209	0.186	0.169	0.156
$L_{contrast}(\gamma=1)$	0.386	0.197	0.147	0.125	0.110	0.100
$0.8L_{cluster}+0.1L_{sim}$	0.491	0.334	0.288	0.263	0.243	0.226
$0.8L_{cluster}+0.1L_{contrast}$	0.503	0.345	0.297	0.271	0.252	0.237
$0.1L_{sim}+0.1L_{contrast}$	0.413	0.247	0.204	0.184	0.170	0.159
$0.8L_{cluster}+0.1L_{sim}+0.1L_{contrast}$(EUVH)	**0.506**	**0.351**	**0.305**	**0.280**	**0.261**	**0.245**

F. Comparison with State-of-the-arts

We compare our EUVH with various state-of-the-art (SOTA) unsupervised hashing methods, including Multiple Feature Hashing (MFH) [10], Deep Hashing (DH) [60], Joint Temporal Appearance Encoder (JTAE) [43], Self-Supervised Temporal Hashing (SSTH) [42], Self-Supervised Video Hashing (SSVH) [15], Unsupervised Hashing method based on Bidirectional Transformers (BTH) [8], Dual-stream Knowledge Preserving Hashing (DKPH) [47] and Multi-granularity Contextualized and Multi-Structure preserved Hashing (MCMSH) [9]. Particularly, the image hashing method DH is extended to video hashing by using the same CNN features. These methods cover different network backbones and data structures. We make comprehensive comparisons and show the results on three datasets with hash code lengths of 16, 32, and 64 bits in Figure 4.

Results on FCVID. Figure 4(a-c) presents a comparison of EUVH and SOTAs on the FCVID dataset. Overall, our EUVH achieves the best performance among all competing methods under all hash code lengths (i.e., 16, 32, and 64 bits). Compared to the transformer-based methods BTH and DKPH, which achieve long-range dependency modeling by multi-head attention mechanism, both MCMSH and EUVH, which can model not only the long-range dependency but also the middle-range and short-range dependencies jointly, significantly improve the performance by a large margin (1.2%-4% under all hash code lengths). This verifies that capturing multiple temporal contexts can benefit video content modeling indeed. In comparison to the strongest competitor MCMSH, EUVH also performs slightly better with respect to short hash code lengths of 16 and 32 bits but has considerable improvement for 64 bits (e.g., 0.335→0.351 mAP@20). The results show that adopting group contextualization can not only reduce the size of the model but also improve performance, proving the effectiveness of our strategy.

Results on ActivityNet. The performance comparison on ActivityNet dataset is shown in Figure 4(d-f). Compared with other SOTAs, EUVH has obvious improvement, e.g., at least 1%, 2.5% and 2.5% for mAP@5 absolute performance improvement on 16-bits, 32-bits, and 64-bits respectively. Compared with FCVID which has ∼91k videos, ActivityNet has only 28k videos. As observed that our EUVH is a lightweight model that has only 1.37M parameters compared to BTH's 3.17M and MCMSH's 1.76M so as to be easily trained to process small datasets. The relatively larger improvements compared with those on FCVID have demonstrated this argumentation. It is also worth noting that the related MCMSH performs relatively worse than DKPH, while EUVH considering similar contexts outperforms DKPH significantly. This in a sense further verifies the effectiveness of group contextualization and structural controlling used by EUVH.

Results on YFCC. Figure 4(g-i) depicts the results of different methods on YFCC dataset. YFCC is a much larger dataset (a million-scale dataset) than FCVID and ActivityNet. We observe a different performance trend compared with those on the small datasets YFCC and ActivityNet. That is, EUVH does not achieve consistently better performance than current SOTAs, e.g., DKPH, MCMSH, and BTH. For the setting of 16 bits, EUVH even performs relatively worse than the current SOTA DKPH on mAP@20-100 with 16 bits. We speculate that this may be because the small size of the model might limit the model capacity. By jointly comparing the results on the three datasets, the performance improvements on small datasets are much more noticeable than that on the large one. However, considering the trade-off between performance and model size, EUVH achieves competitive performance.

Results with different hash code lengths. To clearly show the performance changes with different hash code lengths, we draw the curves of performance changes in Figure 5. It is not surprising that increasing the hash code length can improve the retrieval performance. Because more hash codes can enlarge the capacity of storing video content. But the improvement gradually decreases with hash code length increases. These curves provide us with a clear view to select an appropriate hash code length for a retrieval task.

G. Cross-dataset Performance Comparison

We investigate the generalization of EUVH by conducting across-dataset retrieval that trained on FCVID and tested on YFCC with 64 bits. Table VII displays the performance of mAP@20 of various methods. We can see that all methods have a performance degradation and our EUVH degrades slightly. This is because the FCVID and YFCC have different scenes, quality, duration, etc., which will cause the distribution of the two datasets to be different. Therefore, training on one dataset and testing on another will cause performance degradation. In comparison to MCMSH, our EUVH has less generalization. However, when compared to other approaches, our EUVH has better generalization. Particularly, compared to the transformer-based BTH, the MLP-based MCMSH and EUVH equipped with multiple axial contexts show a better generalization.

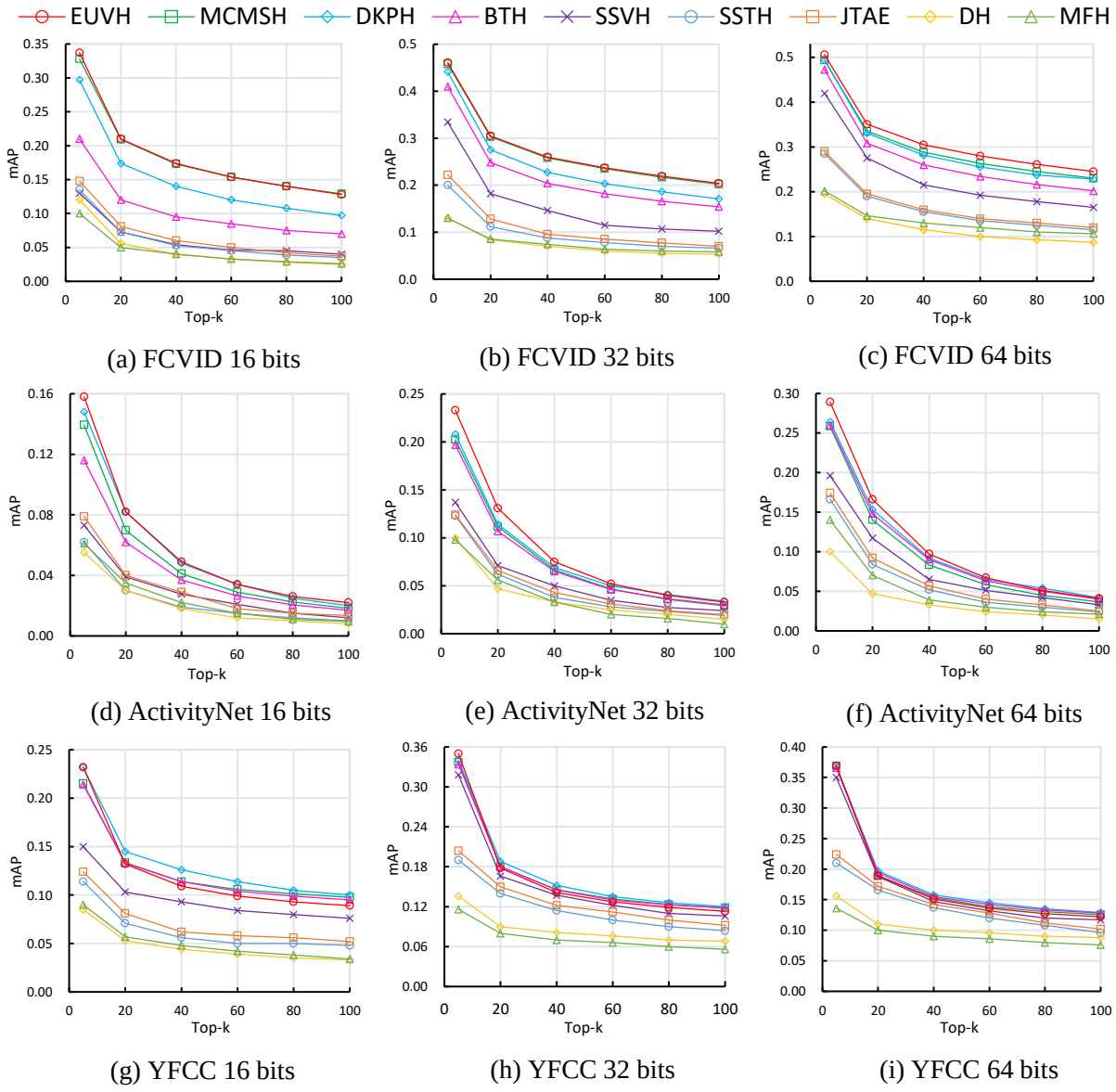


Fig. 4. Performance (mAP@k) comparison with state-of-the-arts on FCVID (a-c), ActivityNet (d-f) and YFCC (g-i) datasets.

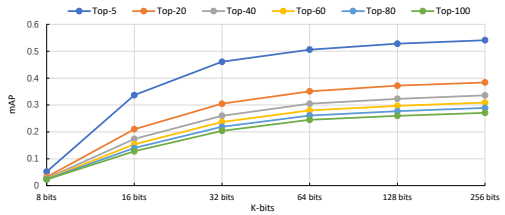
TABLE VII
CROSS-DATASET MAP@20 GAIN WHEN TRAINING ON FCVID AND TEST ON YFCC WITH 64 BITS.

Method	SSTH	SSVH	NPH	BTH	MCMSH	EUVH
mAP@20	$-6.3\% \downarrow$	$-7.8\% \downarrow$	$-6.0\% \downarrow$	$-5.7\% \downarrow$	$-3.2\% \downarrow$	$-3.5\% \downarrow$

H. Qualitative Results

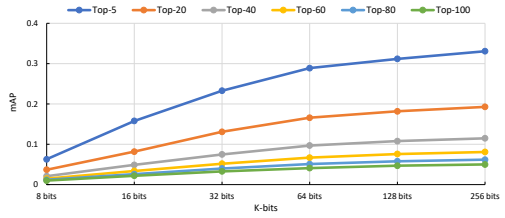
In this section, We further evaluate the proposed EUVH by visualization results, including the visualization for learned hash code distribution and retrieval examples. We first apply t-SNE [61] to project the 64-bit hash codes generated with a single loss ($L_{cluster}$, L_{sim} or $L_{contrast}$) into a 2D space for visualization. As shown in Figure 6, it is consistent with the results listed in Table VI that the *cluster* loss shows better results than the other losses. This again demonstrates the predominance of cluster structure in model learning. Also, we compare the hash code distributions of MCMSH and EUVH in Figure 7. As observed, EUVH shows a better visualization result than MCMSH. By taking the visualization results with a single loss, the combined loss (i.e., $0.8L_{cluster} + 0.1L_{sim} + 0.1L_{contrast}$) shows the most distinguishable distribution, which verifies the strong complementarity of the three structures.

We also visually compare the top-5 retrieved videos of MCMSH and EUVH with 64 bits hash codes on ActivityNet dataset. The results list in Figure 8. Here, we manually select four video categories that describe human-performing actions, e.g., “scuba diving”, “playing drums”, “calf roping”, and “removing ice from car”. Retrieving these actions not only needs recognizing visual objects from the query and



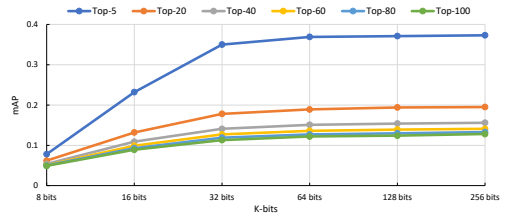
EUVH	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
Top-5	0.052	0.337	0.461	0.506	0.528	0.541
Top-20	0.032	0.210	0.305	0.351	0.372	0.384
Top-40	0.027	0.174	0.260	0.305	0.323	0.336
Top-60	0.025	0.154	0.237	0.280	0.297	0.309
Top-80	0.024	0.140	0.219	0.261	0.277	0.289
Top-100	0.023	0.128	0.204	0.245	0.260	0.271

(a) FCVID



EUVH	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
Top-5	0.063	0.158	0.233	0.289	0.312	0.331
Top-20	0.037	0.082	0.131	0.166	0.182	0.193
Top-40	0.021	0.049	0.075	0.097	0.108	0.115
Top-60	0.015	0.034	0.052	0.067	0.076	0.081
Top-80	0.012	0.026	0.040	0.051	0.058	0.062
Top-100	0.010	0.022	0.033	0.041	0.047	0.050

(b) ActivityNet



EUVH	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
Top-5	0.078	0.232	0.350	0.369	0.371	0.373
Top-20	0.062	0.132	0.178	0.189	0.194	0.195
Top-40	0.054	0.109	0.141	0.151	0.154	0.156
Top-60	0.051	0.099	0.127	0.136	0.139	0.141
Top-80	0.050	0.093	0.119	0.127	0.130	0.133
Top-100	0.049	0.089	0.113	0.122	0.124	0.128

(c) YFCC

Fig. 5. mAP curve with different hash code length (K bits) on FCVID (a), ActivityNet (b) and YFCC (c) dataset.

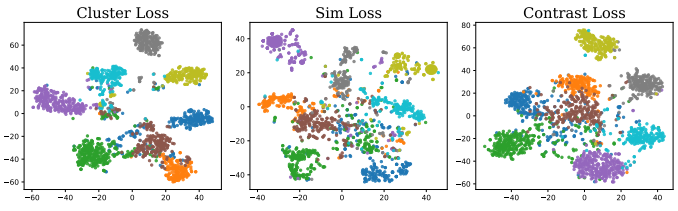


Fig. 6. Visualizing feature distributions of three different structural controlling.

candidates, e.g., “scuba”, “calf”, “drum”, “ice” and “car”, but also requires capturing interactions between human and object or between objects. As shown in the four results, for the “scuba diving” that prefers the static scene rather than the motions, both MCMSH and our EUVH perform well. For the other three activities “playing drums”, “calf roping”,

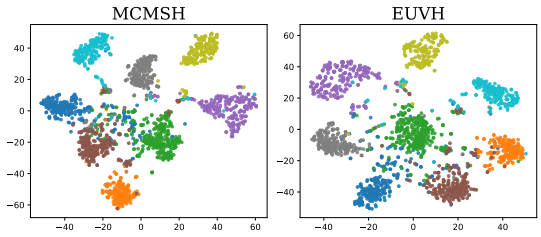


Fig. 7. Visualizing feature distributions of MCMSH and EUVH.

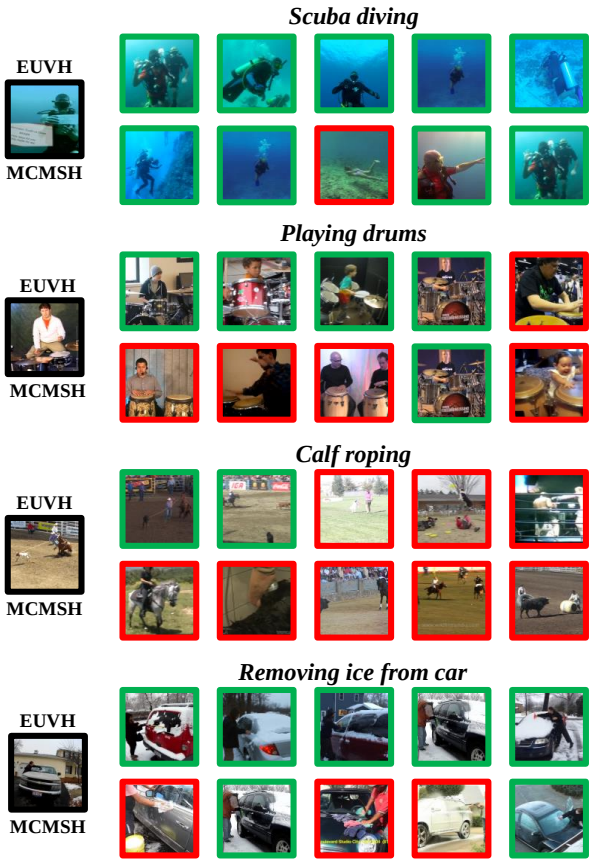


Fig. 8. Top-5 retrieved results of MCMSH and EUVH on ActivityNet with 64 bits codes. Videos in green squares are retrieved correctly, while videos in red squares are wrong.

and “removing ice from car” which focus more on short-term motions/interactions. We can see that both MCMSH and EUVH can model the interactions. This can be attributed to the nature of the multi-context modeling of MCMSH and EUVH. However, most of the top-5 returned results of MCMSH are wrong. For example, MCMSH outputs “horse riding” and “bullfighting” for the “calf roping”, “playing *congas*” rather than “playing *drums*”, and “hand car wash” not “removing ice from car”. While EUVH successfully does it. We guess that it is because EUVH leaves a channel group to not do time contextualization so that it can retain the visual object information largely. In addition, our EUVH also retrieves wrong videos for some queries. For example, for the query “Calf roping” EUVH gets the failed videos showing “Disc dog” or “Powerbocking”. We speculate that these unsuccessful

results can largely be attributed to the limitations of model size and the training dataset. Looking ahead, we envision the potential for enhancing the method's semantic understanding by harnessing both large-scale pretrained vision-language models and large language models (LLMs).

V. Conclusion

This paper has presented an efficient unsupervised video hashing method, EUVH, which not only achieves better or comparable retrieval performance but also reduces the computation cost largely. EUVH splits the video tensor into four groups along the channel dimension and separately refines three of them by three self-gating modules L/M/S-RD in parallel and leaves one group to do nothing. The three self-gating modules focus on feature contextualization by considering different scales of axial contexts, i.e., long-range dependency (L-RD), middle-range dependency (M-RD) and short-range dependency (S-RD). The last non-contextualized one is expected to pay attention to the visual objects appeared in the video. Due to the channel grouping operation, EUVH is much more lightweight than MCMSH with about 22% fewer parameters. In the model training, we proposed to combine three data structures to approximate the real data structure as accurately as possible, including cluster, local neighborhood similarity, and contrast relation. The three structures show different structure preferences, global cohort information, local neighborhood relations, and contrast Information, respectively. We verify our EUVH thoroughly by conducting extensive experiments on three benchmark datasets. Although EUVH achieves satisfactory performance in all experiments, there is still room for improvement. For example, in the current EUVH, we fix a channel group to receive a specific context, so can a channel automatically decide the kind of context or assign variable weights to different contexts. In the future work, we will pay attention to selective feature contextualization.

Acknowledgments

The work was supported by the National Natural Science Foundation of China under Grants No. 9227010114 and No. 62101524.

References

[1] Y. Gu, C. Ma, and J. Yang, "Supervised recurrent hashing for large scale video retrieval," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 272–276.

[2] X. Luo, D. Wu, Z. Ma, C. Chen, M. Deng, J. Ma, Z. Jin, J. Huang, and X.-S. Hua, "Cimon: Towards high-quality hash codes," *arXiv preprint arXiv:2010.07804*, 2020.

[3] N. Han, J. Chen, G. Xiao, H. Zhang, Y. Zeng, and H. Chen, "Fine-grained cross-modal alignment network for text-video retrieval," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3826–3834.

[4] L. Xu, X. Zeng, W. Li, and L. Bai, "Idhashgan: Deep hashing with generative adversarial nets for incomplete data retrieval," *IEEE Transactions on Multimedia*, vol. 24, pp. 534–545, 2022.

[5] M. Su, G. Gu, X. Ren, H. Fu, and Y. Zhao, "Semi-supervised knowledge distillation for cross-modal hashing," *IEEE Transactions on Multimedia*, vol. 25, pp. 662–675, 2023.

[6] H. Lai, P. Yan, X. Shu, Y. Wei, and S. Yan, "Instance-aware hashing for multi-label image retrieval," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2469–2479, 2016.

[7] L. Jin, X. Shu, K. Li, Z. Li, G.-J. Qi, and J. Tang, "Deep ordinal hashing with spatial attention," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2173–2186, 2018.

[8] S. Li, X. Li, J. Lu, and J. Zhou, "Self-supervised video hashing via bidirectional transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 13 549–13 558.

[9] Y. Hao, J. Duan, H. Zhang, B. Zhu, P. Zhou, and X. He, "Unsupervised video hashing with multi-granularity contextualization and multi-structure preservation," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3754–3763.

[10] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 423–432.

[11] Y. Hao, T. Mu, J. Y. Goulermas, J. Jiang, R. Hong, and M. Wang, "Unsupervised t-distributed video hashing and its deep hashing extension," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5531–5544, 2017.

[12] X. Wu, A. G. Hauptmann, and C.-W. Ngo, "Practical elimination of near-duplicates from web video search," in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 218–227.

[13] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 915–928, 2007.

[14] Y. Hao, T. Mu, R. Hong, M. Wang, N. An, and J. Y. Goulermas, "Stochastic multiview hashing for large-scale near-duplicate video retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 1–14, 2016.

[15] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3210–3221, 2018.

[16] S. Li, Z. Chen, J. Lu, X. Li, and J. Zhou, "Neighborhood preserving hashing for scalable video retrieval," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8212–8221.

[17] S. Li, X. Lia, J. Lu, and J. Zhou, "Structure-adaptive neighborhood preserving hashing for scalable video search," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[18] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[19] Y. Hao, H. Zhang, C.-W. Ngo, and X. He, "Group contextualization for video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 928–938.

[20] Y. Li and J. van Gemert, "Deep unsupervised image hashing by maximizing bit entropy," *arXiv preprint arXiv:2012.12334*, 2020.

[21] X. Long, C. Gan, G. De Melo, J. Wu, X. Liu, and S. Wen, "Attention clusters: Purely attention based local feature integration for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7834–7843.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[24] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2916–2929, 2012.

[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2074–2081.

[26] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 37–45.

[27] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2064–2072.

[28] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1522–1530.

[29] Q.-Y. Jiang and W.-J. Li, "Asymmetric deep supervised hashing," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[30] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," *Advances in neural information processing systems*, vol. 21, 2008.

[31] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 18–25.

[32] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2938–2945.

[33] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Icml*, 2011.

[34] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2475–2483.

[35] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 3034–3044, 2018.

[36] L. Cao, Z. Li, Y. Mu, and S.-F. Chang, "Submodular video hashing: a unified framework towards video pooling and indexing," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 299–308.

[37] G. Ye, D. Liu, J. Wang, and S.-F. Chang, "Large-scale video hashing via structure learning," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2272–2279.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[40] V. E. Liong, J. Lu, Y.-P. Tan, and J. Zhou, "Deep video hashing," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1209–1219, 2016.

[41] L. Shen, R. Hong, H. Zhang, X. Tian, and M. Wang, "Video retrieval with similarity-preserving deep temporal hashing," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 4, pp. 1–16, 2019.

[42] H. Zhang, M. Wang, R. Hong, and T.-S. Chua, "Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 781–790.

[43] C. Li, Y. Yang, J. Cao, and Z. Huang, "Jointly modeling static visual appearance and temporal pattern for unsupervised video hashing," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 9–17.

[44] Y. Dong and J. Li, "Video retrieval based on deep convolutional neural network," in *Proceedings of the 3rd International Conference on Multimedia Systems and Signal Processing*, 2018, pp. 12–16.

[45] G. Wu, J. Han, Y. Guo, L. Liu, G. Ding, Q. Ni, and L. Shao, "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2018.

[46] S. Li, Z. Chen, X. Li, J. Lu, and J. Zhou, "Unsupervised variational video hashing with 1d-cnn-lstm networks," *IEEE Transactions on Multimedia*, vol. 22, no. 6, pp. 1542–1554, 2019.

[47] P. Li, H. Xie, J. Ge, L. Zhang, S. Min, and Y. Zhang, "Dual-stream knowledge-preserving hashing for unsupervised video retrieval," in *European Conference on Computer Vision*. Springer, 2022, pp. 181–197.

[48] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[49] D. Hu, F. Nie, and X. Li, "Deep binary reconstruction for cross-modal hashing," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 973–985, 2019.

[50] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5608–5617.

[51] S. Su, Z. Zhong, and C. Zhang, "Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[52] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[53] A. Van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv e-prints*, pp. arXiv–1807, 2018.

[54] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, "Exploiting feature and class relationships in video categorization with regularized deep neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 352–364, 2017.

[55] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 961–970.

[56] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "The new data and new challenges in multimedia research," *arXiv preprint arXiv:1503.01817*, vol. 1, no. 8, 2015.

[57] P. Over, "Trecvid 2013–an overview of the goals, tasks, data, evaluation mechanisms and metrics," 2013.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[60] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[61] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.