Self-supervised Video Hashing via Bidirectional Transformers

Shuyan Li[1,3], Xiu Li[1,3], Jiwen Lu[1,2,*], Jie Zhou[1,2]
[1]Department of Automation, Tsinghua University, China
[2]Beijing National Research Center for Information Science and Technology, China
[3] Graduate School at Shenzhen, Tsinghua University, China
li-sy16@mails.tsinghua.edu.cn; xiu.li@sz.tsinghua.edu.cn;{lujiwen,jzhou}@tsinghua.edu.cn

Abstract

Most existing unsupervised video hashing methods are built on unidirectional models with less reliable training objectives, which underuse the correlations among frames and the similarity structure between videos. To enable efficient scalable video retrieval, we propose a self-supervised video Hashing method based on Bidirectional Transformers (BTH). Based on the encoder-decoder structure of transformers, we design a visual cloze task to fully exploit the bidirectional correlations between frames. To unveil the similarity structure between unlabeled video data, we further develop a similarity reconstruction task by establishing reliable and effective similarity connections in the video space. Furthermore, we develop a cluster assignment task to exploit the structural statistics of the whole dataset such that more discriminative binary codes can be learned. Extensive experiments implemented on three public benchmark datasets, FCVID, ActivityNet and YFCC, demonstrate the superiority of our proposed approach.

1. Introduction

Scalable video retrieval aims at automatically seeking similar videos from a large database related to the content of a query video. Because of the ever-growing abundance of videos from a variety of social media and search engines, developing effective video retrieval technologies has become an urgent need. In order to meet the expectations of efficient scalable retrieval and low storage cost, hashing methods have won lots of interests which transforms high-dimensional data to compact binary codes while preserving the similarity structure between data [3, 5–8, 13, 23, 25, 27, 28, 41, 43]. Among them, learning-based hashing which leverages data properties or label supervision to learn reliable hash functions has achieved promising performance in image retrieval. Compared with images, however, structured data such as videos have not been adequately studied in hashing field [9, 10, 18, 20, 34, 35, 47–49]. For one thing, exploiting the correlations between video frames adds difficulties to video hashing. For another, lack of large-scale labeled dataset such as ImageNet [31] brings further challenges to capture the similarity structure in video data. Therefore, developing effective unsupervised video hashing methods is urgently needed.

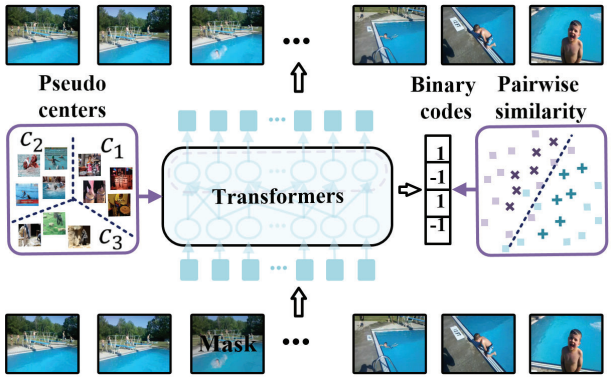


Figure 1: Basic idea of BTH. We design BTH model as bidirectional transformers to better capture the correlations among frames. Then we design a self-supervised learning framework to learn discriminative binary codes.

State-of-the-art unsupervised video hashing approaches focus on exploiting the inherent visual information in each video by using deep neural networks such as Recurrent Neural Networks (RNNs) [1, 17, 19, 35, 35, 49]. Some of them further equip the network with manually constructed similarity guidance [19, 35]. However, most of these methods are built on unidirectional models which underuse the bidirectional correlations among frames. Besides, they fail to adequately exploit the similarity structure in the video data due to suboptimal similarity matrix construction.

In this paper, we propose a self-supervised video hashing method based on bidirectional transformers, which ad-

*Corresponding author

equately exploits the correlations among frames within a video and similarity structure between videos. Unlike most existing video hashing methods which exploit unidirectional correlations among frames, we build our hash model on basis of masked language models to capture the bidirectional correlations [4, 36]. To better unveil the similarity structure among videos, we design a pairwise similarity reconstruction objective. Specifically, we derive a way to establish reliable and effective pairwise similarity connections in the video space. Moreover, we develop a pseudo center set from the training dataset and enforce cluster assignment on the latent outputs of transformers to learn more discriminative hash functions. Figure 1 depicts the basic idea of BTH. We conduct extensive experiments on three widely-used video datasets, FCVID [14], ActivityNet [12], YFCC [38]. The experimental results demonstrate the advantage of BTH over state-of-the-art unsupervised video hashing methods.

2. Related Work

The goal of learning-based video hashing is to learn data-dependent hash functions that yield compact binary codes to achieve good video retrieval accuracy [11]. In an early phase, most works considered image frames within a video separately and indexed videos by simply exploiting conventional hashing methods such as iterative quantization [7] and spectral hashing [43]. A representative method was Multiple Feature based Hashing for video near-duplicate detection (MFH) [34]. However, they failed to employ the spacial structure information such as temporal consistency in videos to design hash functions. Ye *et al.* [47] proposed Video Hashing with both Discriminative commonality and Temporal consistency (VHDT), which was the first work taking the video structure into consideration. Chen *et al.* [2] proposed nonlinear structural hashing by further exploiting the nonlinear relationship between video samples.

In recent years, many deep learning-based hashing methods have been proposed for video retrieval thanks to the great achievement of deep neural networks in extracting high-level semantics. For example, Liong *et al.* [20] proposed Deep Video Hashing (DVH) which passed successive image frames to the convolutional and pooling layers to obtain frame-wise binary representations. Qi *et al.* [37] proposed 3DCNN-based hash, attempting to capture the motion information through multiple successive frames. A major drawback of their method is that 3DCNN can only cope with short video clips of 16 frames [39]. Recently, RNN-based video hashing methods have achieved state-of-the-art performance. For example, Zhang *et al.* [49] proposed Self-Supervised Temporal Hashing (SSTH) based on a binary LSTM autoencoder. Song *et al.* [35] designed a neighborhood similarity loss on top of the hash layer to exploit the neighborhood structure. Wu *et al.* [45] proposed Unsupervised Deep Video Hashing (UDVH) which balanced the variation of each dimension in a binary code. A work similar to theirs [44] was then proposed which replaced the baseline model with Temporal Segment Networks (TSNs) [42]. More recently, Li *et al.*proposed Neighborhood Preserving Hashing (NPH) [19] which preferentially encoded neighborhood-relevant visual content of a video into a binary code referring to pre-extracted neighborhood information. However, all these video hashing methods failed to adequately exploit the bidirectional correlations among frames and similarity structure between videos.

3. Approach

3.1. Bidirectional Transformer Encoder

Given a dataset with N training videos, BTH aims to learn a nonlinear mapping that encode each video to a compact binary vector such that the similarity structure between videos is well preserved. We use CNN features of M sampled frames $\{\boldsymbol{v}_i^m\}_{m=1}^M \in \mathbb{R}^{M\times d}$ to represent a video, where d is the dimension of each frame feature and i is the index of the video. For each video, we feed $\{\boldsymbol{v}_i^m\}_{m=1}^M$ to a bidirectional transformer based hash model to obtain binary codes $\boldsymbol{b}_i \in \{-1, 1\}^k$, where k is the code length.

We design our hash model as bidirectional transformer layers with a hash layer on top of them.

Bidirectional transformer layer. Inspired by the great success of self-attention in capturing correlations in a sequence [40], we adopt bidirectional transformers to capture the correlations among frames within a video. We refer to the multi-layer bidirectional transformer architecture introduced in [40] to build our model. We consider frame features $\{\boldsymbol{v}_i^m\}_{m=1}^M$ to be a sequence of input visual words, which contain important visual details such that visual content can be captured in generated binary codes. Since the module does not exploit the ordering information in the input sequence, we further use positional encoding to remedy the defect. We choose to use sine and cosine functions of different frequencies to encode the position of each frame: $\boldsymbol{p}_{m,2j} = sin(m/10000^{2j/d})$, $\boldsymbol{p}_{m,2j+1} = cos(m/10000^{2j/d})$, where m is the index of the frame and j is the dimension. It yields a sequence of position embeddings, which have equal length and dimension with the input visual sequence. We add these position embeddings to frame features at the bottom of the encoder as input. We present the input sequence in a matrix form $\boldsymbol{X}_i \in \mathbb{R}^{M\times d}$ to detail the computation of the transformer block.

As in a standard transformer block [40], we project the input matrix $\boldsymbol{X}_i$ to three matrices query $\boldsymbol{Q}_i$, key $\boldsymbol{K}_i$ and value $\boldsymbol{V}_i$ via three learnable parameters W^k, W^q and W^v. We use $\boldsymbol{Q}_i$ to perform a scaled dot-product attention over $\boldsymbol{K}_i$, then we put the generation through a softmax function to determine attentional distributions over $\boldsymbol{V}_i$. The resulting

weight-averaged value matrix forms the intermediate output of the transformer block:

$$\boldsymbol{X}_i^{update} = \text{softmax}(\frac{\boldsymbol{X}_i W^q (\boldsymbol{X}_i W^k)^T}{\sqrt{d_k}}) \boldsymbol{X}_i W^v, \quad (1)$$

where d_k is a scaling factor. After being passed through L transformer layers, these input tokens are mapped to a sequence of l-D latent visual embeddings $\{\boldsymbol{h}_i^m\}_{m=1}^M$. Each of these embeddings contains not only the visual content in corresponding frame, but also information flowing from other frames in both direction within the video.

Hash layer. We design a hash layer on top of bidirectional transformers to perform dimension reduction and discretization to these latent latent visual embeddings $\{\boldsymbol{h}_i^m\}_{m=1}^M$. In detail, we project $\{\boldsymbol{h}_i^m\}_{m=1}^M$ to a sequence of real-valued vectors $\{\boldsymbol{t}_i^m\}_{m=1}^M$ via a Fully Connected (FC) layer. The dimension of $\boldsymbol{t}_i^m$ is the same as the code length k. Then we fuse $\{\boldsymbol{t}_i^m\}_{m=1}^M$ via mean pooling as a relaxed binary vector $\bar{\boldsymbol{t}}_i$. We also tried to concatenate $\{\boldsymbol{t}_i^m\}_{m=1}^M$ and map the concatenation to a relaxed binary vector via a FC layer, but we empirically found that the mean pooling achieved better performance. Finally, we discretize $\bar{\boldsymbol{t}}_i$ to a binary vector $\boldsymbol{b}_i$. The resulting binary vector integrates information from all the latent outputs of transformers. We formulate this process as follows:

$$\begin{aligned} &\{\boldsymbol{t}_i^m\}_{m=1}^M = FC(\{\boldsymbol{h}_i^m\}_{m=1}^M, k), \\ &\bar{\boldsymbol{t}}_i = \tanh(\frac{1}{M}\sum_{m=1}^M \boldsymbol{t}_i^m), \quad \boldsymbol{b}_i = \text{sgn}(\bar{\boldsymbol{t}}_i), \end{aligned} \quad (2)$$

where $\text{sgn}(x) = 1$ if $x \geq 0$ and $\text{sgn}(x) = -1$ otherwise. $FC(x_1, x_2)$ is to project a vector x_1 to a x_2-D vector.

3.2. Self-supervised Learning Tasks

In order to learn effective hash functions, we propose these following unsupervised tasks.

Visual cloze task. Inspired by masked language model which enables deep bidirectional representation learning [4, 36], we design a visual cloze task to capture bidirectional correlations between frames. During training stage, we randomly mask p percent of frame features $\{\boldsymbol{v}_i^m\}_{m=1}^M$ from each input video and aim to reconstruct the original frame sequence. Compared with only predicting masked words as previous works do [4, 36], reconstructing the entire frame sequence encourages hash functions to capture more visual content in the video. Specifically, we replace the randomly chosen visual word with (1) an all-zero vector 80% of the time and (2) a randomly chosen frame feature from the training database 10% of the time and (3) the original one 10% of the time. We use [MASK] to denote the masked token and present the input sequence for training as $\boldsymbol{v}_i^1, ..., [\text{MASK}], ..., \boldsymbol{v}_i^M$. We use a FC layer to reconstruct original frame features from $\{\boldsymbol{t}_i^m\}_{m=1}^M$ derived in (2). We present the decoding process as:

$$\{\tilde{\boldsymbol{v}}_i^m\}_{m=1}^M = FC(\{\boldsymbol{t}_i^m\}_{m=1}^M, d), \quad (3)$$

where $\{\tilde{\boldsymbol{v}}_i^m\}_{m=1}^M$ are reconstructed frame features. We use Mean Square Error (MSE) to measure how well these training videos are reconstructed:

$$L_v = \frac{1}{dMN}\sum_{i=1}^N \sum_{m=1}^M ||\boldsymbol{v}_i^m - \tilde{\boldsymbol{v}}_i^m||_2^2. \quad (4)$$

Similarity reconstruction task. While the cloze task can enforce binary codes to capture inherent visual content in each video, it can not explicitly guarantee similarity preserving in the data space. Therefore we further apply pairwise similarity constraint on generated binary codes to enhance similarity preserving.

First of all, we need to establish pairwise similarity connections between videos to guide the similarity preserving training. Since building similarity connections by using all N training videos is time-consuming, we refer to [19] which built an approximate similarity graph $\boldsymbol{S}$ based on a small anchor set $\{\boldsymbol{u}_j\}_{j=1}^n$ ($n \ll N$). Each entry $S_{pq} \in \{-1, 1\}$ ($p, q \in \{1, 2, ..., N\}$) denotes the similarity between the p-th and the q-th training video. We establish the similarity graph in a preprocessing stage as follows.

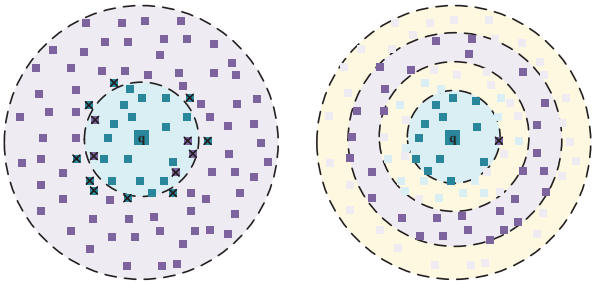
We further train a model which has the same network as BTH with only visual cloze task. We process each training video via this model to obtain a series of vectors from latent outputs $\{\boldsymbol{h}_i^m\}_{m=1}^M$. Then we conduct mean pooling on $\{\boldsymbol{h}_i^m\}_{m=1}^M$ to gain a latent variable $\bar{\boldsymbol{h}}_i \in \mathbb{R}^l$. We consider it as a video feature and we obtain a video feature set over the training database $\{\bar{\boldsymbol{h}}_i\}_{i=1}^N$. We conduct K-means clustering on $\{\bar{\boldsymbol{h}}_i\}_{i=1}^N$ to obtain a anchor set $\{\boldsymbol{u}_j\}_{j=1}^n$, where each anchor is a clustering center. For each $\bar{\boldsymbol{h}}_i$, we calculate a nearest anchors $\boldsymbol{u}_{i1}, \boldsymbol{u}_{i2}, ..., \boldsymbol{u}_{ia}$, which are denoted as $\{\boldsymbol{u}_{i*}\}$ ($i* \in [1, n]$) for sake of brevity. Then we derive a truncated similarity matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times n}$:

$$Y_{ij} = \begin{cases} \frac{\exp(-Dist(\bar{\boldsymbol{h}}_i, \boldsymbol{u}_j)/t)}{\sum_{j'=1}^a \exp(-Dist(\bar{\boldsymbol{h}}_i, \boldsymbol{u}_{ij'})/t)}, \forall \boldsymbol{u}_j \in \{\boldsymbol{u}_{i*}\} \\ 0, \quad \text{otherwise} \end{cases} \quad (5)$$

where $Dist()$ is a distance calculation function. As [30] indicates that such embedding distance can be naturally measured in squared Euclidean space, we adopt l_2-norm as the specific form of $Dist()$. t is a bandwidth parameter. Then an approximate adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is calculated as:

$$\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{\Lambda}^{-1}\boldsymbol{Y}^T, \quad (6)$$

where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{Y}^T\boldsymbol{1}) \in \mathbb{R}^{n \times n}$. We set $S_{pq} = 1$ if $A_{pq} > 0$ and $S_{pq} = -1$ otherwise.



(a) (b)

Figure 2: Schematic illustration of established similarity structure. For a training video, we select positive samples in blue area and negative samples in purple area. We leave out samples in yellow areas. Our proposed similarity construction strategy in (b) is more reliable and effective compared with (a). Here, blue/purple dots denotes samples that are ground-truth similar/dissimilar.

We consider the distances between vectors as proxy indicators of their semantic similarity as [15] did. We illustrate the established pairwise similarity given a query video in Figure 2(a). As can be seen, such unsupervised similarity construction method tends to make mistakes around the boundary surface, which makes the similarity matrix less reliable. Besides, while there are large amount of negative samples, those around the boundary surface are most helpful for learning discriminative distance metric. In order to derive more reliable and effective pairwise similarity, we make the following adjustments. We set the number of nearest anchor a as $a_1, a_2, a_3 (a_1 < a_2 < a_3)$ respectively, and calculate three intermediate similarity matrices $\boldsymbol{S}^1, \boldsymbol{S}^2, \boldsymbol{S}^3$ accordingly. We calculate each entry of the final similarity matrix $\boldsymbol{S}$ as:

$$S_{pq} = \begin{cases} 1, & S^1_{pq} = 1 \\ -1, & S^2_{pq} = -1 \text{ and } S^3_{pq} = 1 \\ 0, & otherwise \end{cases} \tag{7}$$

We show the final pairwise similarity structure in Figure 2(b). We only consider training pairs when $S_{pq} \in \{1, -1\}$, which improves the reliability and effectiveness of similarity learning. During training, 50% of the time we randomly sample two videos which are considered to be similar ($S_{pq} = 1$), and 50% of the time we randomly sample two videos which are considered to be dissimilar ($S_{pq} = -1$).

We compute the similarity between two binary vectors as $s(\boldsymbol{b}_p, \boldsymbol{b}_q) = \frac{1}{k}\boldsymbol{b}_p^T \boldsymbol{b}_q$. We aim to approximate $s(\boldsymbol{b}_p, \boldsymbol{b}_q)$ to the established pairwise similarity. As the optimization is intractable due to sgn function, we instead use the relaxed binary vector $\bar{\boldsymbol{t}}$ in (2) and formulate the quantization error as a penalty term. We design the similarity reconstruction loss as:

$$L_s = \frac{1}{N} \sum_{(p,q)\sim\mathcal{P}} (S_{pq} - \frac{1}{k}\bar{\boldsymbol{t}}_p^T \bar{\boldsymbol{t}}_q)^2 + ||\boldsymbol{b}_p - \bar{\boldsymbol{t}}_q||_2^2. \tag{8}$$

where $\mathcal{P}$ denotes the video pair sampling strategy on the training set.

Cluster alignment task. A major drawback of unsupervised hash learning is that the intrinsic structure of the whole sample space can be skewed within training batches due to lack of label information [22, 24, 32]. To learn more discriminative binary codes, we further exploit the structural statistics of the whole training dataset. We reuse the anchor set $\{\boldsymbol{u}_j\}_{j=1}^n$ described in above section, where each entry represents a pseudo clustering center in the training set. Given the i-th video data point, we assign it with a nearest clustering center $\boldsymbol{u}_{i1}$ according to the l_2-norm distance between $\bar{\boldsymbol{h}}_i$ and each center. We consider $\{\boldsymbol{u}_{i1}\}$ as pseudo labels derived from the training data, where correlations among these labels are reflected from the distances between clustering centers. During training, we approximate $\bar{\boldsymbol{h}}_i$ to the corresponding center $\boldsymbol{u}_{i1}$. This procedure intuitively endows the model with more informative latent semantics. We expect to optimize the clustering loss:

$$L_c = \frac{1}{N} \sum_{i=1}^{N} ||\bar{\boldsymbol{h}}_i - \boldsymbol{u}_{i1}||_2^2. \tag{9}$$

Overall loss. The overall loss to optimise is given by:

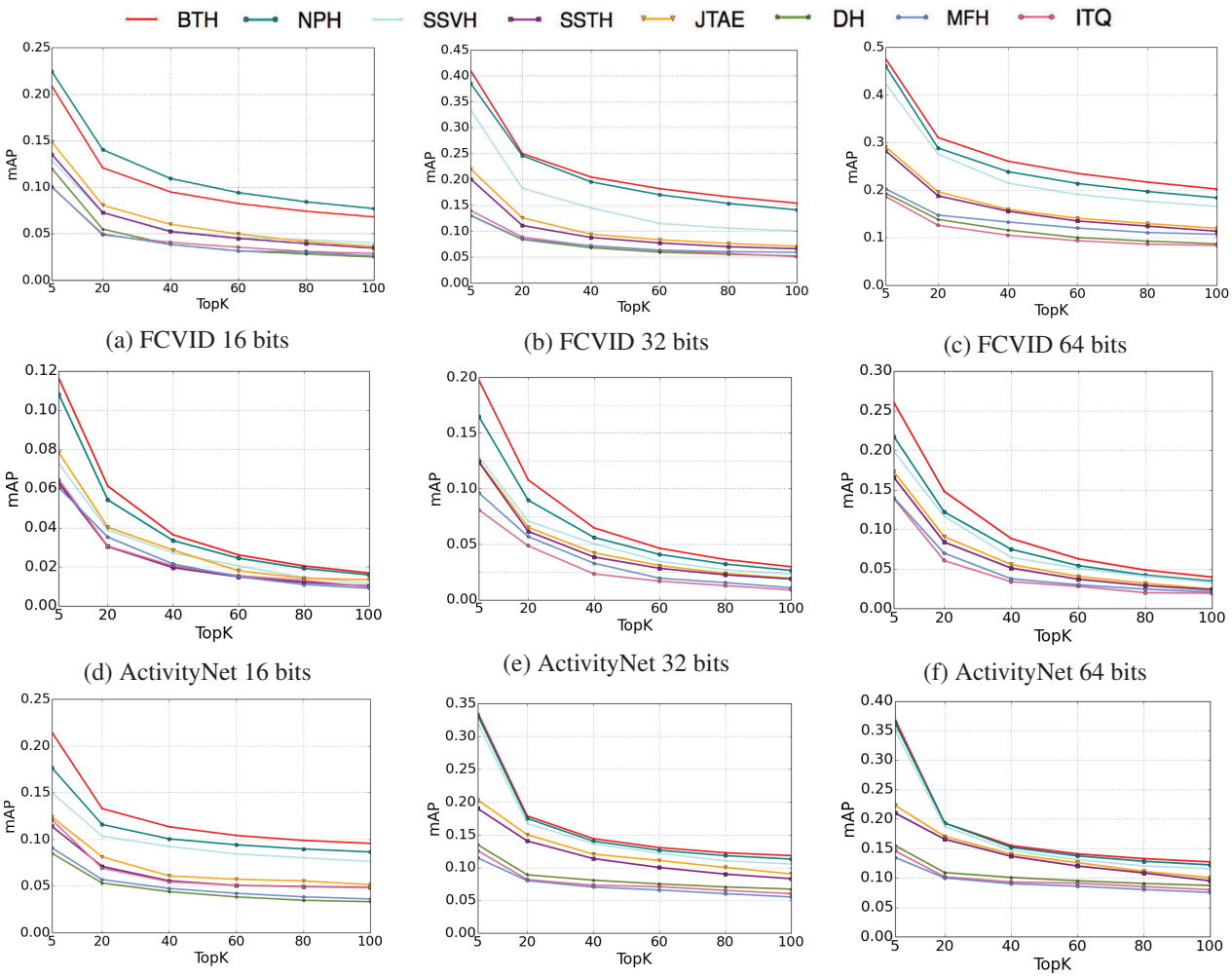
$$L = \alpha_1 L_v + \alpha_2 L_s + \alpha_3 L_c, \tag{10}$$

where α_1, α_2, and α_3 are hyper-parameters that balance above-mentioned three losses. As we apply constrains on real-valued vectors instead of binary codes and use quantization error as a penalty term, we directly use back-propagation to optimize the whole network.

3.3. Implementation Details

For sake of training efficiency, we only built the similarity matrix $\boldsymbol{S}$ and pseudo centers $\{\boldsymbol{u}_j\}_{j=1}^n$ in the initialization stage. We also tried to update them based on the updating of video features $\{\bar{\boldsymbol{h}}_i\}$ during training, however, we found it did not bring significant improvement on the performance. As we adopted BTH to obtain $\{\bar{\boldsymbol{h}}_i\}$ in the preprocessing stage, the amount of information contained in $\{\bar{\boldsymbol{h}}_i\}$ heavily depended on the dimension of hash bottleneck layer k. We found that when k is too small, these video features could not capture enough visual information to provide reliable training guidance. Therefore we set k as a relatively large value, 128, in the preprocessing stage. We fixed $\boldsymbol{S}$ and $\{\boldsymbol{u}_j\}_{j=1}^n$ during hash learning.

Similar to SSTH [49], for each video, we uniformly sampled 25 frames. We used VGG-16 network [33] pre-trained



(a) FCVID 16 bits (b) FCVID 32 bits (c) FCVID 64 bits

(d) ActivityNet 16 bits (e) ActivityNet 32 bits (f) ActivityNet 64 bits

(g) YFCC 16 bits (h) YFCC 32 bits (i) YFCC 64 bits

Figure 3: Retrieval performances among all hashing methods in terms of mAP@K over three datasets.

on ImageNet [31] to extract 4096-D frame features. For FCVID and YFCC, we directly used the features which were kindly provided by Zhang *et al.*[1]. We kept the input frame features the same for all the compared state-of-the-art methods for fair comparison. We used single layer transformer with 256-D single attention head as our bidirectional transformer encoder. We set the size of pseudo center set as 2000 and we conducted 10 iterations for K-means clustering to obtain it. We set the scaling factor d^k as 256. In the training phase, we empirically set the values of α_1, α_2 and α_3 as 0.9,0.1 and 0.8 respectively to achieve good performance. We set the masking rate p as 15% as [26] did. we set the batch size as 128 and the initial learning rate as 0.0005. We firstly trained our model with only visual cloze task for 50 epochs and finetuned it with full loss for another 50 epochs. We optimized our model by using Adam optimizer algorithm [16] with momentum 0.9. We implemented our model by using Pytorch[2]. Our model is publicly available at https://github.com/Lily1994/BTH.

4. Experimental Results

4.1. Dataset and Evaluation Protocols

Dataset: We evaluated our proposed method on these following three datasets: FCVID [14], ActivityNet [12] and YFCC [38]. **FCVID** comprises 91,223 videos which are manually annotated into 239 categories. The categories cover a variety of topics such as activities, objects, events, etc. Same as [49], we utilized 91,185 videos of them, with 45,585 videos for training and 45,600 videos as retrieval database and queries. **ActivityNet** involves a wide range of human activities in 200 categories. We used 9,722 videos

[1] https://pan.baidu.com/s/1i65ccHv

[2] http://pytorch.org/

as training set and exploited validation set as our test set since the test split was not publicly available. In each category, we randomly selected 1,000 videos for queries and 3,760 videos for retrieval database. **YFCC** is a huge video dataset which composes of 0.8M videos. We used 409,788 unlabeled data for training and 101,256 labeled data for out-of-sample retrieval test. The labeled videos were collected from the third level of MIT SUN scene hierarchy [46], which involved 80 categories. Among videos with none-zero labels, we randomly picked 1000 videos as queries and the remaining ones as retrieval database.

Evaluation protocols: Following the same evaluation protocol as [49], we utilized mean Average Precision at top-K retrieved results (mAP@K) to evaluate the retrieval performance [29]. Besides, we used Precision-Recall (PR) curve as an additional evaluation metric. We sorted the retrieved results according to Hamming ranking. We evaluated the performance with code lengths of 16, 32, and 64 bits.

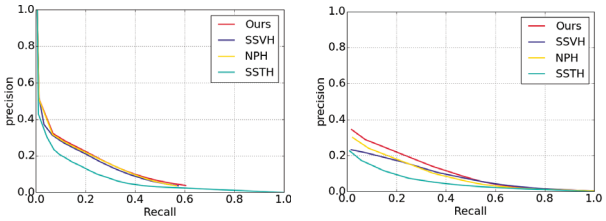
4.2. Results and Analysis

Comparisons with state-of-the-arts: We compared BTH with several state-of-the-art unsupervised hashing methods to demonstrate the superior performance of it: ITQ [7], DH [21], MFH [34], SSTH [49], JTAE [17], SSVH [35] and NPH [19]. We extended the image hashing methods ITQ and DH to video hashing by implementing them on CNN frame features as [35] did. We reported the mAP@K results on FCVID, ActivityNet and YFCC datasets in Figure 3.

As shown in Figure 3(a)-(c), our proposed BTH outperforms all these compared hashing methods except for NPH on FCVID with all code lengths remarkably. As can be seen, BTH outperforms the most competitive NPH with code lengths of 32 bits and 64 bits. Compared with NPH, the mAP@K(K=20,60,100) values of BTH are 2.3%, 2.2% and 1.9% higher respectively with 64 bits. We owe this performance advantage to the bidirectional transformer module which effectively captures the bidirectional correlations among frames and elaborately designed self-supervised learning tasks. It is worth to mention that NPH requires pre-established anchor set in retrieval stage while our method does not have such requirement. This makes our proposed method more efficient and practical for realistic retrieval systems.

As shown in Figure 3(d)-(f), BTH significantly outperforms the other methods with all code lengths. For example, it outperforms the most competitive NPH by 3.3% and 4.3% with 32 bits and 64 bits respectively in terms of mAP@5.

As shown in Figure 3(g)-(i), BTH consistently outperforms the other methods with all code lengths, which shows its superiority on this dataset. The performance advantage over the strongest competitor NPH is most significant with



(a) FCVID 64 bits (b) ActivityNet 64 bits

Figure 4: PR curves of different video hashing methods.

code length of 16 bits on this dataset. In terms of mAP@5, BTH outperforms NPH by 4.5% with 16 bits.

The PR curves of BTH, NPH, SSVH and SSTH with 64 bits on FCVID and ActivityNet are shown in Figure 4. As can be seen, BTH achieves higher precision than all these compared state-of-the-arts at the same rate of recall on both datasets. When recall rate is allowed to be low, the precision advantage of BTH becomes more significant.

Ablation study: Firstly, to show the superiority of the bidirectional transformers, we substitute transformers with LSTM, BiLSTM and CNN respectively and compare BTH with these three baselines on FCVID in Table 1. It shows that BTH outperforms these three baselines significantly with 32 bits and 64 bits, which validates the advantage of transformers over LSTM, BiLSTM and CNN. Here, transformers form the attention between any two frames in the video in parallel, which is beneficial to model correlations between distant frames. This is the major advantage over sequential models such as LSTM and BiLSTM, which cannot well preserve long-term dependencies in the video. In addition, limited receptive field makes CNN focus more on local information, which cannot well capture correlations between distant frames either.

Secondly, we evaluated the effectiveness of the visual cloze task by comparing our method with these following baselines. 1) BTH-L_v. We removed the visual reconstruction loss L_v during training. 2) BTH-mask1. We only reconstructed the masked visual words instead of the whole input sequence. 3) BTH-mask2. During training, we used the original frame feature sequence instead of masked one as input. As shown in Table 1, the absence of L_v will heavily deteriorate the retrieval accuracy, which indicates the importance of visual content reconstruction. Besides, only predicting the masked words can not achieve good retrieval accuracy. This indicates that this training strategy cannot make binary codes capture adequate visual content. Furthermore, masking some of the input visual words can further improve the performance, which indicates that the masked model is beneficial to capture bidirectional correlations among frames.

Then we evaluated the effectiveness of the similarity reconstruction task by comparing our method with these fol-

Table 1: mAP@K results with different baseline models on FCVID with 32-bit codes and 64-bit codes.

Methods	32bits					64bits				
	K=20	K=40	K=60	K=80	K=100	K=20	K=40	K=60	K=80	K=100
CNN	0.242	0.197	0.172	0.155	0.140	0.287	0.233	0.202	0.180	0.164
LSTM	0.239	0.195	0.170	0.153	0.139	0.284	0.229	0.200	0.179	0.163
BiLSTM	0.245	0.200	0.174	0.158	0.142	0.291	0.235	0.205	0.182	0.166
BTH-L_v	0.218	0.174	0.152	0.136	0.124	0.276	0.228	0.203	0.185	0.172
BTH-mask1	0.228	0.184	0.163	0.142	0.133	0.287	0.239	0.213	0.195	0.181
BTH-mask2	0.240	0.196	0.174	0.148	0.146	0.298	0.252	0.226	0.208	0.194
BTH-L_s1	0.229	0.186	0.162	0.145	0.133	0.287	0.238	0.212	0.193	0.180
BTH-L_s2	0.238	0.194	0.171	0.155	0.142	0.296	0.248	0.222	0.204	0.190
BTH-L_c1	0.218	0.171	0.149	0.133	0.121	0.272	0.224	0.198	0.180	0.166
BTH-L_c2	0.230	0.183	0.161	0.145	0.133	0.284	0.236	0.210	0.191	0.178
BTH-L_c3	0.238	0.192	0.169	0.155	0.143	0.292	0.245	0.221	0.204	0.191
BTH	**0.248**	**0.204**	**0.182**	**0.166**	**0.154**	**0.308**	**0.260**	**0.234**	**0.216**	**0.202**

lowing baselines. 1) BTH-L_s1. We removed the similarity reconstruction loss L_s during training. 2) BTH-L_s2. We directly used the similarity matrix derived in [19] to guide the training. As shown in Table 1, the retrieval performance drops when L_s is missing, which indicates that capturing the pairwise similarity is important to learn effective binary codes. In addition, that BTH outperforms BTH-L_s2 indicates that our proposed pairwise similarity connections are beneficial to unveil the similarity structure between videos.

To evaluate the effectiveness of the cluster alignment task, we compared our model with following proposed baselines. 1) BTH-L_c1. We removed the clustering loss L_c during training. 2) BTH-L_c2. We randomly built the pseudo center set and the pseudo labels for each video were obtained by nearest neighbor search from this set. 2) BTH-L_c3. We set the dimension of BTH bottleneck layer as 64 (insetad of 128 which was used in this paper) and used $\{\bar{\boldsymbol{h}}_i\}$ produced by it to establish the pseudo center set. As shown in Table 1, the missing of L_c deteriorates the retrieval accuracy, which indicates that exploiting the structural statistics of the whole training dataset can lead to significant performance improvement. Moreover, we can observe that BTH-L_c2 outperforms BTH-L_c1 but fails to achieve equal retrieval accuracy with BTH. This indicates that assigned pseudo labels can help to capture the similarity structure in the whole dataset even though the pseudo center set is randomly built. Meanwhile, the quality of the pseudo center set has great impact on the retrieval performance and the unsupervised construction method adopted in this paper is beneficial to learn more discriminative binary codes. Furthermore, that BTH outperforms BTH-L_c3 indicates that when the dimension of the bottleneck layer is small, the obtained video features cannot capture enough visual information to construct reliable pseudo center set.

We also tried to update the pseudo center set based on the updating of video features $\{\bar{\boldsymbol{h}}_i\}$ during training. We report the update of mAP@5 and mAP@60 on FCVID with 64 bits in Figure 5. Here Stage1 denotes that the model is trained with only cloze task. Stage2 denotes the model is fine-tuned with full loss where the pseudo center set is established in the initialization stage. Stage3 denotes the model is further fine-tuned with the pseudo center set calculated by updated $\{\bar{\boldsymbol{h}}_i\}$. As can be seen, updating the pseudo center set does not bring significant improvement to mAP@60 and even slightly deteriorates mAP@5. Therefore, we do not further update the pseudo center set during training.

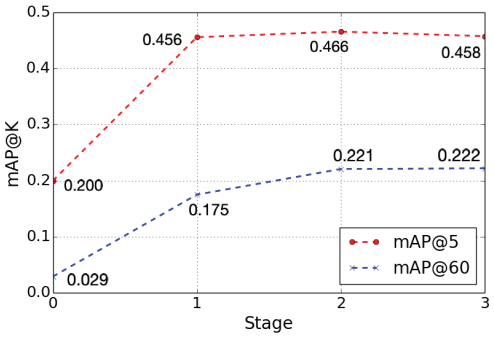


Figure 5: Update of mAP@5 and mAP@60 during training on FCVID with 64 bits.

Cross-dataset evaluation comparisons: We investigated how BTH generalizes to cross-dataset retrieval by training various methods on FCVID and test on YFCC. We compared the retrieval performance with single-dataset case (both training and test on YFCC). TABLE 2 shows mAP@20 results of various methods with 64 bits in cross-dataset setting. While the performances of all these methods

Table 2: Cross-dataset mAP@20 of various methods when training on FCVID and test on YFCC with 64 bits. Red number indicates the performance drop compared with training and testing both on YFCC.

Method	SSTH	SSVH	NPH	BTH
mAP@20	0.155↓ 6.3	0.173↓ 7.8	0.180↓ 6.0	0.191↓ 5.7

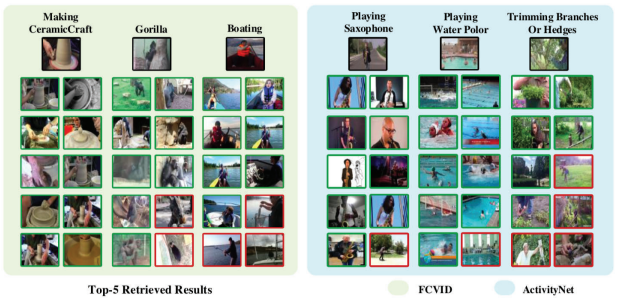


Figure 6: Examples of top-5 retrieved results. The first row are queries, below which the left columns are retrieved videos by BTH and right ones are retrieved by SSVH. Green borders denote correct and red borders denote incorrect.

decrease compared with single-dataset case, BTH still outperforms the other methods. Besides, the performance drop of BTH is less than the other methods, which indicates its better generalization cross different datasets.

Qualitative results: We showed top-5 retrieved results with 64 bits on FCVID and ActivityNet datasets by BTH and SSVH in Figure 6. While both methods are able to provide relative candidates in top of the retrieved ones, BTH consistently retrieves more correct videos. For example, given a query video in category "Gorilla", BTH returns five correct videos while SSVH retrieves two incorrect videos which describe other animals. Furthermore, as for the category "Trimming Branches Or Hedges", the retrieved videos look very similar since the background is full of grass. It is crucial to exploit the correlations among frames to better understand the action in this case. It shows that BTH works better under this circumstance.

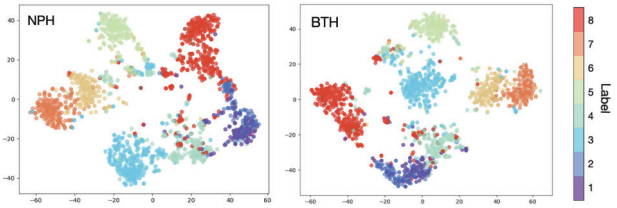


Figure 7: t-SNE visualizations of BTH and NPH. Videos are randomly sampled on FCVID database, and samples with different labels are marked with different colors.

Table 3: Encoding time of various deep hash methods.

Methods	SSTH	SSVH	NPH	BTH
Encoding time	0.88ms	1.03ms	1.42ms	1.18ms

We showed the t-SNE visualizations of BTH and the most competitive NPH on FCVID with 64 bits in Figure 7. The target is to cluster the video points with the same label and separate the video points with different labels. We can observe that hash codes generated by BTH in different categories are better separated, which indicates that BTH can generate more discriminative binary codes that NPH.

Encoding time: The time cost to generate binary codes for query videos is crucial to evaluate the practical retrieval system. We reported the encoding time, time to encode frame features of a video to binary codes, of BTH, NPH, SSTH and SSVH in Table 3. We kept the implementation platforms the same. We can observe that BTH has nearly the same time complexity with SSTH and SSVH, while it outperforms these two methods significantly. This indicates that BTH is practical for retrieval systems. Besides, BTH has less time complexity while achieving higher accuracy most of the cases than NPH.

5. Conclusion

In this work, we proposed a self-supervised hashing method, BTH, for scalable video retrieval. BTH efficiently captured correlations among frames via bidirectional transformers to learn discriminative hash functions. Based on the encoder-decoder structure of transformers, we designed three self-supervised learning tasks to adequately capture the similarity structure in video data: A visual cloze task which reconstructed original input frame sequence based on masked inputs; A similarity reconstruction task which enforced the similarity of generated binary vector pair consistent with the pre-established pairwise similarity; A cluster alignment task that enforced the latent embedding aligned with pseudo clustering centers. Experiments on three widely-used benchmark datasets demonstrated the superiority of our proposed method over state-of-the-arts. In the future, we can try different frame sampling strategies to further improve the retrieval performance.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, the National Natural Science Foundation of China under Grants 41876098, U1813218, 61822603, U1713214, 61672306, 61572271, and 61527808.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *TNNLS*, 5(2):157–166, 1994.

[2] Zhixiang Chen, Jiwen Lu, Jianjiang Feng, and Jie Zhou. Nonlinear structural hashing for scalable video search. *TCSVT*, 28(6):1421–1433, 2018.

[3] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *CVPR*, pages 6838–6847, June 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

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

[6] Yueqi Duan, Jiwen Lu, Jianjiang Feng, and Jie Zhou. Context-aware local binary feature learning for face recognition. *TPAMI*, 40(5):1139–1153, 2018.

[7] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization-a procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2013.

[8] Yuchen Guo, Guiguang Ding, and Jungong Han. Robust quantization for general similarity search. *TIP*, 27(2):949–963, Feb 2018.

[9] Yanbin Hao, Tingting Mu, John Yannis Goulermas, Jianguo Jiang, Richang Hong, and Meng Wang. Unsupervised t-distributed video hashing and its deep hashing extension. *TIP*, 26(11):5531–5544, 2017.

[10] Yanbin Hao, Tingting Mu, Richang Hong, Meng Wang, Ning An, and John Yannis Goulermas. Stochastic multiview hashing for large-scale near-duplicate video retrieval. *TMM*, 19(1):1–14, 2017.

[11] Junfeng He, Shih-Fu Chang, Regunathan Radhakrishnan, and Claus Bauer. Compact hashing with joint optimization of search accuracy and time. In *CVPR*, pages 753–760, 2010.

[12] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015.

[13] Y. Jiang, J. Wang, X. Xue, and S. Chang. Query-adaptive image search with hash codes. *TMM*, 15(2):442–453, 2013.

[14] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *TPAMI*, 40(2):352–364, 2018.

[15] Vani K., Sandra Mitrovic, Alessandro Antonucci, and Fabio Rinaldi. SST-BERT at semeval-2020 task 1: Semantic shift tracing by clustering in bert-based embedding spaces. *CoRR*, abs/2010.00857, 2020.

[16] Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[17] Chao Li, Yang Yang, Jiewei Cao, and Zi Huang. Jointly modeling static visual appearance and temporal pattern for unsupervised video hashing. In *CIKM*, pages 9–17, 2017.

[18] Shuyan Li, Zhixiang Chen, Xiu Li, Jiwen Lu, and Jie Zhou. Unsupervised variational video hashing with 1d-cnn-lstm networks. *TMM*, 22(6):1542–1554, 2020.

[19] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. Neighborhood preserving hashing for scalable video retrieval. In *ICCV*, pages 8211–8220, 2019.

[20] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, and Jie Zhou. Deep video hashing. *TMM*, 19(6):1209–1219, 2017.

[21] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.

[22] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[23] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[24] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[25] Xianglong Liu, Lei Huang, Cheng Deng, Bo Lang, and Dacheng Tao. Query-adaptive hash code ranking for large-scale multi-view visual search. *TIP*, 25(10):4514–4524, 2016.

[26] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *CoRR*, abs/1908.02265, 2019.

[27] Yueming Lv, Wing W. Y. Ng, Ziqian Zeng, Daniel S. Yeung, and Patrick P. K. Chan. Asymmetric cyclical hashing for large scale image retrieval. *TMM*, 17(8):1225–1235, 2015.

[28] Jonathan Masci, Michael M. Bronstein, Alexander M. Bronstein, and Jürgen Schmidhuber. Multimodal similarity-preserving hashing. *TPAMI*, 36(4):824–830, 2014.

[29] Paul Over, George Awad, Jonathan G. Fiscus, Brian Antonishek, Martial Michel, Wessel Kraaij, Alan F. Smeaton, and Georges Quénot. TRECVID 2010 - an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2014.

[30] Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B. Viégas, Andy Coenen, Adam Pearce, and Been Kim. Visualizing and measuring the geometry of BERT. In *NIPS*, pages 8592–8600, 2019.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[32] Yuming Shen, Li Liu, and Ling Shao. Unsupervised binary representation learning with deep variational networks. *IJCV*, 127(11-12):1614–1628, 2019.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[34] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, pages 423–432, 2011.

[35] Jingkuan Song, Hanwang Zhang, Xiangpeng Li, Lianli Gao, Meng Wang, and Richang Hong. Self-supervised

video hashing with hierarchical binary auto-encoder. *TIP*, 27(7):3210–3221, 2018.

[36] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. *CoRR*, abs/1904.01766, 2019.

[37] Jiande Sun, Haifeng Qi, Jing Li, Wenbo Wan, and Qiang Wu. A 3D-CNN based video hashing method. *ICDIP*, page 82, 08 2018.

[38] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *CoRR*, abs/1503.01817, 2015.

[39] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. In *CVPR*, pages 4489–4497, 2015.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[41] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *TPAMI*, 34(12):2393–2406, 2012.

[42] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *CoRR*, abs/1705.02953, 2017.

[43] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, volume 282, pages 1753–1760. 2009.

[44] Gengshen Wu, Jungong Han, Yuchen Guo, Li Liu, Guiguang Ding, Qiang Ni, and Ling Shao. Unsupervised deep video hashing via balanced code for large-scale video retrieval. *TIP*, 28(4):1993–2007, 2019.

[45] Gengshen Wu, Li Liu, Yuchen Guo, Guiguang Ding, Jungong Han, Jialie Shen, and Ling Shao. Unsupervised deep video hashing with balanced rotation. In *IJCAI*, pages 3076–3082, 2017.

[46] J. Xiao, J. Hays, K. A Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010.

[47] Guangnan Ye, Dong Liu, Jun Wang, and Shih-Fu Chang. Large-scale video hashing via structure learning. In *CVPR*, pages 2272–2279, 2013.

[48] Litao Yu, Zi Huang, Jiewei Cao, and Heng Tao Shen. Scalable video event retrieval by visual state binary embedding. *TMM*, 18(8):1590–1603, 2016.

[49] Hanwang Zhang, Meng Wang, Richang Hong, and Tat-Seng Chua. Play and rewind: optimizing binary representations of videos by self-supervised temporal hashing. In *ACM MM*, pages 781–790, 2016.