Structure-adaptive Neighborhood Preserving Hashing for Scalable Video Search

Shuyan Li, Xiu Li, Jiwen Lu, *Senior Member, IEEE*, and Jie Zhou, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a Structure-adaptive Neighborhood Preserving Hashing (SNPH) method for unsupervised scalable video search. Unlike most existing hashing methods which equally encode an entire video into a binary feature vector, we propose a neighborhood attention mechanism which encodes the neighborhood-relevant content of a video to better preserve the neighborhood relationships among videos. Motivated by the fact that a video usually contains multiple shots and each shot depicts a different activity, we further develop a structure-adaptive encoder to model the hierarchical structure of the video. Specifically, the encoder adaptively divides each video into multiple segments via detecting temporal boundaries across frames and encodes these segments as a compact binary vector to capture rich structural information. We integrate the neighborhood attention mechanism into the structure-adaptive encoder to learn hash functions that jointly preserve the neighborhood relationships among videos and exploit the hierarchical structure in a video. Experimental results on three widely used benchmark datasets show that our proposed method consistently outperforms state-of-the-art unsupervised video hashing methods.**

***Index Terms*—hashing, video search, structure-adaptive, neighborhood preserving, unsupervised.**

I. Introduction

Over the past decade, we have witnessed a rapid increase of video data on the Internet. For example, there are more than 1.8 billion videos on the YouTube video website with around 300 hours of videos being uploaded per minute. Meanwhile, a variety of video applications such as SnackVideo and Tik Tok have sprung up like mushrooms, resulting in explosive growth of online video data. The explosion of video data makes fast video search extremely important, however, conventional nearest neighbor search methods using exhaustive linear scanning are infeasible due to tremendous computational complexity and storage requirement [1]. To enable efficient scalable search, hashing-based approximate nearest neighbor search approach, which transforms high-dimensional data to compact binary codes, has been developed and soon become a popular tool [2]–[18]. While existing hashing approaches have achieved promising performance in image search, only a few methods are designed to tackle scalable video search [19]–[25].

Compared with images, videos contain special structure such as temporal consistency and scene shift. It has been shown that exploiting the structure information can improve the expressive capability of video representations [26]–[28]. Early video hashing methods attempted to exploit the structure information via applying similarity constrains across frames [19], [29]. However, they considered a video a set of independent frames and ignored the temporal order of the frame sequence, which inevitably led to suboptimal binary codes. More recently, the success in deep neural network for representation learning has inspired lots of video hashing algorithms [25], [30]–[35]. These deep learning based hashing methods have shown great potential for extracting the temporal information in videos and achieved state-of-the-art in video search. In general, they equally compress an entire video into a binary vector via deep neural networks connected with a hash layer. Since videos usually contain superfluous and even ambiguous content which may interfere with nearest neighbor search, absorbing entire content from a video will lead to unsatisfactory hash functions [36]. On the other hand, none of these methods consider the hierarchical structure of videos, i.e., a video usually contains multiple shots which depict different activities [37]. For example, in Fig. 1, there are three shots in the input video which successively depict "whipping the cream", "folding the sponge cake" and "buttering". Based on the content in these shots, one can conclude more structured information such as "making cake". We argue that ignoring such hierarchical structure will hinder the network from better understanding the content of a video and limit the search performance.

To address the aforementioned issues, we propose a Structure-adaptive Neighborhood Preserving Hashing (SNPH) method for scalable video search in an unsupervised manner. For each video, we derive a neighborhood representation, a vector representation of a pseudo neighbor or integration of several pseudo neighbors of the video. We embed the neighborhood representation into the SNPH encoder to facilitate neighborhood preserving encoding. Specifically, we propose a neighborhood attention mechanism that focuses on partial useful content in each input frame instead of equally encoding the entire content, guided by the neighborhood representation. We further develop a structure-adaptive encoding network which captures the hierarchical temporal structure

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. *(Corresponding author: Jiwen Lu)*

Shuyan Li and Xiu Li are with the Tsinghua Shenzhen International Graduate School, Shenzhen, 518055, China. E-mail: li-sy16@mails.tsinghua.edu.cn; li.xiu@sz.tsinghua.edu.cn.

Jiwen Lu and Jie Zhou are with the Department of Automation, State Key Lab of Intelligent Technologies and Systems, Beijing Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing, 100084, China. E-mail: lujiwen@tsinghua.edu.cn; jzhou@tsinghua.edu.cn.

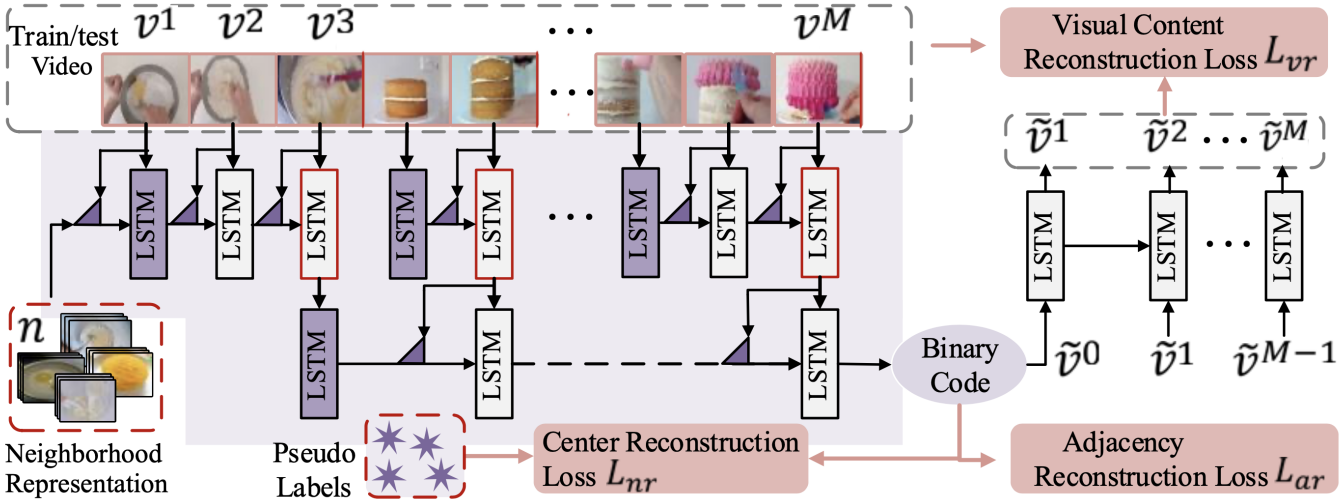


Fig. 1: The flowchart of SNPH. The neighborhood representation is embedded into the encoder to guide neighborhood preserving encoding. The SNPH encoder is in a hierarchical recurrent architecture. The encoder first detects temporal boundaries across frames to adaptively divide a video into several segments and independently encodes them to latent representations. Then it encodes these latent representations of segments as a compact binary vector. A decoder reconstructs the frame features from the binary vector. Triangle denotes the neighborhood attention mechanism. Purple LSTM rectangle denotes initializing/reinitializing LSTM. Red lines denote time boundaries across frames and LSTM in red box means that a time boundary is detected. We omit the Binarization, neighborhood representation calculation, and frame feature extraction for brevity.

in the video to transform each video to a binary vector. In detail, it adaptively divides each long-range video into several segments via automatically identifying time boundaries across frames, and then encodes these segments as a binary vector. We integrate the neighborhood attention mechanism into the structure-adaptive encoding network such that the hierarchical structure of the video is exploited to learn neighborhood preserving hash functions. Furthermore, we develop a pseudo label set, where each point in this set has the same dimension as code length, for self-supervised training. As the pseudo label space is of high dispersion, we enforce the binary vector to approximate the corresponding pseudo label such that a more discriminative Hamming space can be learned. We design SNPH in an encoder-decoder framework, where the flowchart is detailed in Fig. 1. Experimental results on three public benchmark datasets show the effectiveness of our proposed SNPH compared with the state-of-the-arts. We briefly highlight the contributions of this work as follows:

1) We embed a pre-extracted neighborhood representation into the encoder such that it guides to encode useful content in the video to facilitate neighborhood preserving. Specifically, we develop a neighborhood attention mechanism that focuses on partial useful content from each input frame, conditioned on the neighborhood representation.

2) We design a structure-adaptive encoding network for hash learning to leverage the hierarchical temporal structure of the video. The structure-adaptive architecture can efficiently capture the temporal dependencies both intra-activity and inter-activities via detecting time boundaries. We integrate the neighborhood attention mechanism into the encoding network to learn neighborhood preserving hash functions.

3) The experimental results on three real-world datasets show that our proposed SNPH significantly outperforms the state-of-the-art unsupervised video hashing methods.

This paper is the extension of our previous conference paper [34], but it differs in several aspects from that work. While a video may consists of multiple shots which depict different activities, NPH fails to exploit such hierarchical structure of the video. We design a structure-adaptive neighborhood preserving encoder based on two-layer LSTMs, which automatically detects time boundaries in each video, to replace the single-layer LSTM based NPH encoder. The proposed encoder adaptively divides each video into several segments and encodes them into a binary vector, such that the temporal dependencies both within activity and across activities is better captured. To learn more discriminative hash functions, we further develop a pseudo label set for self-supervised training where the dimension of each pseudo label is the same as the code length. We enforce the binary vector to approximate the corresponding pseudo label, such that the intra-class distance is minimized and inter-class distance is maximized in the Hamming space. Experimental results verify the efficacy of our proposed method.

II. Related Work

In this section we briefly review existing content-based video search methods and representative video representation learning methods.

A. Content-based Video Search

Content-based video search aims to retrieve relevant videos for a query video. Since a video is usually represented by a sequence of high-dimensional frame features, directly

calculating the distance in the high-dimensional video space is of high computational complexity [38]–[41]. To reduce the computation cost of linear scan, Approximate Nearest Neighbor (ANN) search methods were proposed [42]–[44]. Among these ANN methods, hashing which projects a video to a low-dimensional binary vector efficiently reduces the computation and storage cost, and is widely applied to scalable video search [6], [19], [21]–[24], [35], [45].

For example, Ye *et al.* [19] exploited the spatial information within each frame and temporal consistency between frames in a video to design hash functions. Chen *et al.* [19] explored the scene structure by minimizing the distances between frame-level binary vectors within the same scene. Hao *et al.* [21] exploited Student t-distribution to estimate the similarity between hash vectors of video frames. Li *et al.* [46] and Qiao *et al.* [47] focused on face video search where the query was usually a face image. The above-mentioned methods can be classified as supervised hashing. Generally speaking, supervised paradigms can achieve higher performance than unsupervised ones, whereas they are less practical for large-scale database due to the huge time- and labor- cost for labeling.

Unsupervised video hashing methods utilize data properties instead of labels to learn discriminative hash functions. Recently, deep neural networks have been applied to learn hash functions in an unsupervised manner and have achieved state-of-the-art performance [25], [31]–[33], [48], [49]. Among them, Zhang *et al.* [31], Song *et al.* [32] and Li *et al.* [33] proposed to learn hash functions via minimizing the feature reconstruction error. Wu *et al.* [48], [49] learned balanced features by minimizing quantization distortion when mapping video features to a binary hypercube. Li *et al.* [25] designed a variational auto-encoder to generate binary codes. Whereas these methods have two weaknesses. On one hand, they ignore the fact that a video usually contains multiple shots which depict different activities, thus under-use the hierarchical structure in the video. On the other hand, they equally exploit entire content from the video, which is likely to absorb irrelevant information and results in performance degradation.

B. Video Representation Learning

In an early stage, quite a few works aggregated frame-level CNN features [50] with different fusion strategies to represent videos [22], [51]–[53]. A major drawback of these methods is that they neglect the order of frame sequences and under-use the temporal structure of the video. Then Simonyan *et al.* [54] proposed two-stream ConvNets to simultaneously exploit the optical flow and appearance information. Inspired by [55], [56], Tran *et al.* [57] proposed 3D ConvNets to generate video representation. However, they are only capable to capture temporal information in short video clips.

Inspired by the success of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [58], [59] in sequence to sequence learning [60], [60]–[62], Ng *et al.* [26] exploited LSTM to learn the temporal order in videos. Venugopalan *et al.* [63] then proposed stacked LSTM to improve non-linearity for better performance. However, it has been shown that LSTM works well only when videos are 30 to 80 frames long [26], [63]. To efficiently model longer video clips, Pan *et al.* [64] proposed hierarchical recurrent video encoder to dramatically shorten the input path. Song *et al.* [32] extended it to self-supervised video hashing and achieved promising search performance. They manually divided a long video clip into several short frame segments as input. However, such hand-crafted partition is hard to reveal the temporal structure and an inaccurate partitions may cause information chaos [65]. To solve this problem, Baraldi *et al.* [37] proposed a special LSTM cell which was able to identify discontinuity points between frames. We integrate such LSTM cell into our model to better explore and leverage the temporal structure of the video.

III. Approach

A. Overview

Given a collection of N videos, SNPH aims to learn a deep hash function $\mathcal{F}$ that encodes each video into a k-bit binary vector $\boldsymbol{b} \in \{-1, 1\}^k$ while preserving neighborhood relationships among videos. Instead of equally absorbing entire content from the video, we embed a pre-extracted neighborhood representation $\boldsymbol{n}$ into the encoder to provide guidance for neighborhood preserving encoding. We present the encoding process as follows:

$$\boldsymbol{b} = \mathcal{F}(\boldsymbol{S}, \boldsymbol{n}, \Theta), \tag{1}$$

where Θ is the parameter set of the SNPH encoding network, namely structure-adaptive neighborhood preserving encoder. The input video $\boldsymbol{S}$ is represented in the form of frame features $\{\boldsymbol{v}^f\}_{f=1}^{M} \in \mathbb{R}^{M \times l}$ extracted by a conventional Convolution Neural Network (CNN). M is the length of the input frame sequence and l is the dimension of the frame feature.

To formulate the structure-adaptive neighborhood preserving hashing, we firstly present the neighborhood attention mechanism and the calculation of neighborhood representation in III-B. Then we describe the structure-adaptive neighborhood preserving encoder in III-C. Finally, we introduce the objective functions in III-D to realise the neighborhood preserving learning.

B. Neighborhood Attention Mechanism

We design a neighborhood attention mechanism to preferentially extract useful content from each input frame conditioned on a pre-extracted neighborhood representation, inspired by a memory state introduced in [66]. In recurrent neural networks, the vector memory state $m^t \in \mathbb{R}^b$ updates by attending over the former memory state m^{t-1} and the current input, where t denotes the time step. Specifically, m^t absorbs the content from current input frame feature $\boldsymbol{v}^t$ and updates as follows:

$$m^t = \text{softmax}\left(\frac{m^{t-1}W^q([m^{t-1}; \boldsymbol{v}^t]W^k)^T}{\sqrt{d^k}}\right)[m^{t-1}; \boldsymbol{v}^t]W^v. \tag{2}$$

Here d^k denotes a scaling factor, W^* is a learnable weight matrix and $[x_1; x_2]$ denotes row-wise concatenation of two tensors x_1 and x_2. The memory states are randomly initialised, i.e., m^0 is set as a random vector. (2) is considered to be an extension

of self-attention [67]: $A(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$. The query $\boldsymbol{Q}$ is to perform a scaled dot-product attention over the keys $\boldsymbol{K}$. The product is then transformed to a set of weights via softmax-function to obtain a weighted average value from $\boldsymbol{V}$. Here, $\boldsymbol{K}$ and $\boldsymbol{V}$ are calculated from $[m^{t-1};\boldsymbol{v}^t]$, and $\boldsymbol{Q}$ is calculated from m^{t-1} [67]. In this way, this self-attention mechanism can learn to write content from input frame into the memory state based on the information already contained in the memory.

The self-attention mechanism only refers to the inherent information in the video, which makes the learned hash functions less effective for nearest neighbor search. In order to better exploit neighborhood relationships among videos, we extend the self-attention mechanism to neighborhood attention mechanism via embedding a pre-extracted neighborhood representation into the memory state. Given an input video $\boldsymbol{S}$, we extract a neighborhood representation $\boldsymbol{n} \in \mathbb{R}^b$ which is assumed to deliver adequate information in neighbors of the video. We incorporate $\boldsymbol{n}$ into the memory state to guide the encoding. While there are a variety of ways to incorporate $\boldsymbol{n}$ into the memory, we choose to inject it into the memory state at the first time-step:

$$m^1 = \text{softmax}\left(\frac{\boldsymbol{n}W^q([\boldsymbol{n};\boldsymbol{v}^1]W^k)^T}{\sqrt{d^k}}\right)[\boldsymbol{n};\boldsymbol{v}^1]W^v. \tag{3}$$

When $t > 1$, the memory state updates with (2). As the information in neighbor videos has been injected into the memory, at each time-step, it interacts with the new input frame feature and guide to incorporate relevant content into the memory state.

Neighborhood representation: Given a small anchor set which contain n anchors $\{\boldsymbol{u}_j^*\}_{j=1}^n$, we design the neighborhood representation $\boldsymbol{n}_i$ for an input video $\boldsymbol{S}_i$ as a pseudo neighbor or an integration of pseudo neighbors retrieved from $\{\boldsymbol{u}_j^*\}_{j=1}^n$ ($i \in [1,N]$ denotes the index of the video in the database). An intuition is that the neighborhood representations calculated from similar videos tend to be similar, hence they can guide to encode similar videos to similar binary vectors.

Among a variety of ways to establish the anchor set $\{\boldsymbol{u}_j^*\}_{j=1}^n$, we conduct clustering in video feature space such that these anchor points have a strong representation power to adequately cover the vast video data points [68]. It allows each anchor point to have equal opportunity to be retrieved as neighbor such that more discriminative neighborhood representations can be obtained. Specifically, we use a pre-trained LSTM autoencoder [27] to extract a video feature $\boldsymbol{u}_i \in \mathbb{R}^b$ for each video, which has the same dimension with the memory state. We use $\{\boldsymbol{u}_i\}_{i=1}^N$ to denote video features of the training database. We conduct K-means clustering on $\{\boldsymbol{u}_i\}_{i=1}^N$ to obtain n clustering centers $\{\boldsymbol{u}_j^*\}_{j=1}^n$. We consider $\{\boldsymbol{u}_j^*\}_{j=1}^n$ to be an anchor set. For an input video $\boldsymbol{S}_i$, we search a nearest neighbors by sorting $l2$-norm distances between $\boldsymbol{u}_i$ and all anchors in $\{\boldsymbol{u}_j^*\}_{j=1}^n$. We use $\boldsymbol{u}_{i1}^*, \boldsymbol{u}_{i2}^*, ..., \boldsymbol{u}_{ia}^*(i1, i2, ..., ia \in \{1,2,...,n\})$ to denote the a nearest anchors. We concatenate them, which are sorted in ascending order according to their distances from $\boldsymbol{u}_i$, in a row-wise manner. We then linearly project the concatenation to obtain the neighborhood representation $\boldsymbol{n}_i \in \mathbb{R}^k$:

$$\boldsymbol{n}_i = W^{nu}[\boldsymbol{u}_{i1}^*; \boldsymbol{u}_{i2}^*; ..; \boldsymbol{u}_{ia}^*] + r_n, \tag{4}$$

where r_* denotes a learnable bias.

It is noteworthy that we only build $\{\boldsymbol{u}_j^*\}_{j=1}^n$ in a preprocessing stage such that it can be reused for future training and Hamming search. While we could also construct the anchor set using the latent outputs during the learning of SNPH detailed in Section III-C, we empirically found that it does not bring significant improvement on the final performance. Considering that the calculation of $\boldsymbol{n}_i$ is also required during test stage, we set $a \ll n \ll N$, such that calculating a nearest anchors from a small anchor set brings negligible extra time cost. Meanwhile the anchor set does not require much storage space, therefore our method is practical for search systems.

C. Structure-adaptive Neighborhood Preserving Encoder

To better capture the temporal dependencies both intra-activity and inter-activities, we design a structure-adaptive neighborhood preserving encoder (SNPH encoder), which is inspired by boundary-aware LSTM network [37]. The SNPH encoder consists of a two-layer LSTM network integrated with neighborhood attention mechanisms and a hash layer. In the first recurrent layer, it automatically identifies time boundaries between frames and adaptively divides a long-range input video into several segments. It independently encodes each segment to a latent representation to avoid mixture of information in different segments. In the second recurrent layer, it projects these segment representations to a high-dimensional real-valued latent output such that the hierarchical structure of the video can be exploited. Finally, the hash layer maps this high-dimensional real-valued latent output to a compact binary vector.

The core of SNPH encoder is a boundary detector d that is able to identify time boundaries between frames. We use $h^{x,t}$ and $c^{x,t} \in \mathbb{R}^b$ to denote the hidden state and the cell state of the x-th recurrent layer at time step t respectively. At each time step, the detector $d^t \in \{0,1\}$ is computed based on the current input frame $\boldsymbol{v}^t$ and hidden state of the first recurrent layer $h^{1,t-1}$ as follows:

$$\begin{aligned} &d^t = \tau(\sigma(W^{dv}\boldsymbol{v}^t + W^{dh}h^{1,t-1} + r_d)), \\ &\tau(x) = \begin{cases} 1, & if \quad x \geq 0.5 \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \tag{5}$$

where σ denotes sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$. $d^t = 1$ means that a time boundary is detected, then we feed $h^{1,t-1}$ to the second recurrent layer. Meanwhile we reinitialize the memory cell and hidden state of the first recurrent layer before next time-step update. $d^t = 0$ means that no time boundary is detected, then the information keeps passing through the first layer. We formulate this process as follows:

$$c^{1,t-1} \leftarrow c^{1,t-1} \cdot (1 - d^t), \tag{6}$$

$$h^{1,t-1} \leftarrow h^{1,t-1} \cdot (1 - d^t). \tag{7}$$

We formulate the updates of both recurrent layers as follows:

$$i^{x,t} = \sigma(W^{iv}\boldsymbol{x}^{x,t} + W^{ih}h^{x,t-1}), \quad (8)$$
$$f^{x,t} = \sigma(W^{fv}\boldsymbol{x}^{x,t} + W^{fh}h^{x,t-1}), \quad (9)$$
$$c^{x,t} = BN(f^{x,t} \odot c^{x,t-1} + i^{x,t} \odot \text{MLP}(\boldsymbol{x}^{x,t}), \quad (10)$$
$$h^{x,t} = \tanh(c^{x,t}), \quad (11)$$

where MLP and BN denote multiple layers perceptron and batch normalization respectively. Here, $\boldsymbol{x}^{x,t}$ denotes the current input of the x-th recurrent layer. For the first layer, $\boldsymbol{x}^{1,t}$ is equal to the frame feature $\boldsymbol{v}^t$, and for the second layer, $\boldsymbol{x}^{2,t}$ is equal to $d^t h^{1,t-1}$. $tanh$ is a hyperbolic tangent function: $tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. With time boundaries detected in the first recurrent layer, each video is adaptively divided into s segments ($s < M$ is a positive integer depending on the input video) which are then independently encoded as latent representations. The latent representations of these segments are passed to the second recurrent layer to extract the hierarchical structure of the video.

Now we integrate the neighborhood attention mechanism proposed in Section III-B into the hierarchical recurrent layers by substituting (10) with:

$$c^{x,t} = BN(f^{x,t} \odot c^{x,t-1} + i^{x,t} \odot \text{MLP}(m^{x,t}). \quad (12)$$

Here, the vector memory $m^{x,t}$ replaces the input vector $\boldsymbol{x}^{x,t}$ in a conventional recurrent layer such that irrelevant content in the input frame can be suppressed.

We initialize the memory state of neighborhood attention mechanism in the first recurrent layer $m^{1,t}$ with the neighborhood representation $\boldsymbol{n}$ with (3), meanwhile we randomly initialize the memory state in the second recurrent layer $m^{2,t}$. As we expect to independently encode each segment in the first recurrent layer, we need to reinitialize $m^{1,t}$ once a time boundary is detected such that the content in previous segments has no impact on later encoding. We use the neighborhood representation to reinitialize $m^{1,t}$ such that it can provide guidance all through the encoding phase. That is, we initialize/reinitialize $m^{1,t}$ before the next time-step update as follows:

$$m^{1,t-1} \leftarrow \begin{cases} \boldsymbol{n}, & if\ t = 1 \\ d^t\boldsymbol{n} + (1 - d^t)m^{1,t-1}, & t > 1. \end{cases} \quad (13)$$

After obtaining the latent output from the last hidden state of the second recurrent layer $h^{2,s}$, we feed it to the hash layer to obtain a single binary vector:

$$\boldsymbol{t} = \tanh(W^{th}h^{2,s} + r), \quad (14)$$
$$\boldsymbol{b} = sgn(\boldsymbol{t}), \quad (15)$$

where $sgn(x) = 1$ if $x \geq 0$ and $sgn(x) = -1$ otherwise. $\boldsymbol{t} \in (-1, 1)^k$ is considered to be a relaxed binary vector. In this way, the SNPH encoder is able to capture the hierarchical structure of the input video and learn neighborhood preserving hash functions.

D. Objective Functions

To enable neighborhood preserving hash learning, we design our model in an encoder-decoder scheme with several objective functions to reconstruct specific information from the relaxed binary vector. 1) We design an LSTM decoder to reconstruct the input frames to ensure that the hierarchical structure of the input video is well captured. We train our model to minimize the distortion between input frame features and reconstructed ones, which is formulated as a visual content reconstruction loss L_{vr}. 2) We build an adjacency matrix in the video feature space, aiming to reconstruct this adjacency matrix in the learned Hamming space. We train our model to minimize the discrepancy between the similarity of each video feature pair and the similarity of each binary vector pair, which is formulated as an adjacency reconstruction loss L_{ar}. 3) To learn more discriminative hash functions, we further develop a pseudo label set. We enforce each binary vector to approximate the corresponding pseudo label such that the intra-class distance is minimized and inter-class distance is maximized in the Hamming space. Accordingly, we design a center reconstruction loss L_{nr} to describe the distance between the pseudo label and the relaxed binary vector. Our loss function L is the weighted sum of these three losses:

$$L = \alpha_1 L_{vr} + \alpha_2 L_{ar} + \alpha_3 L_{nr}, \quad (16)$$

where α_1, α_2 and α_3 are hyper-parameters to balance these three loss terms.

Visual content reconstruction loss: To encourage the binary vectors to well capture the hierarchical structure of videos, we utilize an LSTM decoder to recurrently reconstruct the frame features from the relaxed binary codes. Specifically, we project the relaxed binary vector $\boldsymbol{t}_i$ to a l-D vector $\tilde{\boldsymbol{v}}_i^0$ and consider it the only information provider for the decoder. We inject $\tilde{\boldsymbol{v}}_i^0$ into the decoder at first time step, then obtain the first reconstructed frame feature $\tilde{\boldsymbol{v}}_i^1 \in \mathbb{R}^l$ from the output of the decoder. Afterwards, we feed the current output to the decoder to obtain next reconstructed frame feature recurrently. After M time steps, we obtain M reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^f\}_{f=1}^M$. The decoding process can be presented as:

$$\{\tilde{\boldsymbol{v}}_i^f\}_{f=1}^M = \mathcal{D}(\boldsymbol{t}_i, \Psi), \quad (17)$$

where Ψ is the parameter set of the LSTM decoder. We minimize the discrepancy between reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^f\}_{f=1}^M$ and the input frame features $\{\boldsymbol{v}_i^f\}_{f=1}^M$. We use Mean Square Error (MSE) to formulate the visual content reconstruction loss L_{vr} as:

$$L_{vr} = \frac{1}{lMN}\sum_{i=1}^{N}\sum_{f=1}^{M}||\boldsymbol{v}_i^f - \tilde{\boldsymbol{v}}_i^f||_2^2. \quad (18)$$

Adjacency reconstruction loss. To calculate the adjacency reconstruction loss L_{ar}, we first establish an adjacency matrix $A \in \mathbb{R}^{N \times N}$ which reflects the neighborhood relationships in the video feature space $\{\boldsymbol{y}_i\}_{i=1}^N \in \mathbb{R}^b$. Each entry of it $A_{ij} \in \{-1, 1\}$ denotes the similarity between two training videos $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ (i and $j \in \{1, 2, ..., N\}$). The intuition is that an optimized feature space can reflect semantic clustering even though no label is utilized for training. Therefore we

can extract the pairwise similarity between videos in this feature space as an auxiliary target for self-supervised training. However, the time complexity of directly building a large adjacency matrix is very high given very large N. To enable efficient training, we refer to [68] that first builds a truncated adjacency matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times n}$ by using a small anchor set $\{y_p^*\}_{p=1}^n$ ($n << N$). Similarities of all N video points are measured with respect to these n anchors, and the adjacency matrix $\boldsymbol{A}$ is approximated using these similarities, which greatly reduces the time complexity.

Specifically, we conduct K-means clustering on $\{\boldsymbol{y}_i\}_{i=1}^N$ to get an anchor graph $\{\boldsymbol{y}_p^*\}_{p=1}^n$ which can adequately represent the training videos. For each video $\boldsymbol{S}_i$, we calculate its e nearest anchors $\boldsymbol{y}_{i1}^*, \boldsymbol{y}_{i2}^*, ..., \boldsymbol{y}_{ie}^*$ ($i1, i2, ie \in [1, n]$). Then we obtain a truncated similarity matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times n}$, where each entry denotes the similarity between a video feature $\boldsymbol{y}_i$ and an anchor point $\boldsymbol{y}_p^*$:

$$Y_{ip} = \begin{cases} \frac{\exp\left(-l2(\boldsymbol{y}_i, \boldsymbol{y}_p^*)/bw\right)}{\sum_{p'=1}^{e}\exp\left(-l2(\boldsymbol{y}_i, \boldsymbol{y}_{ip'}^*)/bw\right)}, \forall p \in \langle i \rangle \\ 0, \quad \text{otherwise} \end{cases} \tag{19}$$

where $\langle i \rangle$ denotes the indices of e nearest anchors of $\boldsymbol{y}_i$. bw is a bandwidth parameter and $l2$ denotes $l2$-norm distance. Then we calculate a non-negative and sparse approximate adjacency matrix $\boldsymbol{A}^{approx}$ based on $\boldsymbol{Y}$:

$$\boldsymbol{A}^{approx} = \boldsymbol{Y}\boldsymbol{\Lambda}^{-1}\boldsymbol{Y}^T, \tag{20}$$

where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{Y}^T\boldsymbol{1}) \in \mathbb{R}^{n\times n}$. We set each entry of the final adjacency matrix $A_{ij} = 1$ if the (i, j)-th entry of the approximate adjacency matrix $A_{ij}^{approx} > 0$ and $A_{ij} = -1$ otherwise.

We define the similarity between two binary vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ as $\frac{1}{k}\boldsymbol{b}_i^T\boldsymbol{b}_j$. For steady training, we instead use an approximate similarity $\frac{1}{k}\boldsymbol{t}_i^T\boldsymbol{t}_j$, where $\boldsymbol{t}_i$ is the relaxed binary vector calculated with (14). We use Mean Square Error (MSE) to formulate the discrepancy between the similarity of each video feature pair and the similarity of each binary vector pair. Meanwhile, we apply a quantization error with regard to $\boldsymbol{b}_i$ and $\boldsymbol{t}_i$ as a penalty term. We have the specific form of adjacency reconstruction loss L_{ar} for (16):

$$L_{ar} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(A_{ij} - \frac{1}{k}\boldsymbol{t}_i^T\boldsymbol{t}_j)^2 + \frac{1}{kN}\sum_{i=1}^{N}||\boldsymbol{b}_i - \boldsymbol{t}_i||_2^2. \tag{21}$$

Another intuition is that with video features being optimized during training, more convincing guidance can be provided by an updated $\boldsymbol{A}$. This is inspired by [69]–[71] which iteratively bootstrap the outputs of a network to serve as targets for an enhanced representation. In this paper, we consider video features to be latent outputs of SNPH encoder which update during training. We use these features to construct and update the adjacency matrix, in other words, we use the adjacency matrix derived from the previous epoch to guide the current training. At a beginning step, we train our SNPH model with only visual content reconstruction loss L_{vr} in (18). The memory states of the neighboring attention mechanisms are all randomly initialised. Given the i-th video, we extract the latent output from last hidden state of the second recurrent layer $h_i^{2,s}$ as $\boldsymbol{y}_i$ with (11). We establish $\boldsymbol{A}$ based on $\{\boldsymbol{y}_i\}_{i=1}^N$. Now that we have obtained $\boldsymbol{A}$, in later steps we train our model with full loss L in (16) and iteratively bootstrap $h_i^{2,s}$ to update $\{\boldsymbol{y}_i\}_{i=1}^N$ and $\boldsymbol{A}$ to learn an enhanced binary feature vector. We repeat this process until convergence.

Center reconstruction loss: To learn more discriminative hash functions, we further develop a pseudo label set where the dimension of each pseudo label is the same as the code length. We enforce the binary vector to approximate the corresponding pseudo label. Specifically, we use a pre-trained autoencoder to extract a k-D vector representation for each video. It is noteworthy that we can re-use the SNPH encoder trained for L_{ar} calculation in the beginning step as described in previous section, such that we need not further train an autoencoder. We use this encoder to extract a k-D vector $\boldsymbol{t}_i'$ for each training video V_i with (14). We conduct K-means clustering on $\{\boldsymbol{t}_i'\}_{i=1}^N$ to obtain q clustering centers $\{\boldsymbol{t}_j^*\}_{j=1}^q$ ($q \ll N$). The building of $\{\boldsymbol{t}_j^*\}_{j=1}^q$ is only needed in a pre-processing stage, which will not bring extra time cost for future training and Hamming search.

We consider the center point $\boldsymbol{t}_{i1}^*$ ($i1 \in \{1, 2, .., q\}$) which is closest to $\boldsymbol{t}_i'$ among $\{\boldsymbol{t}_j^*\}_{j=1}^q$ to be the pseudo label of the i-th video. We directly minimize the discrepancy between $\boldsymbol{t}_{i1}^*$ and the relaxed binary vector $\boldsymbol{t}_i$, such that the intra-class distance is minimized and inter-class distance is maximized in the Hamming space. We formulate the center reconstruction loss L_{nr} with MSE:

$$L_{nr} = \frac{1}{Nk}\sum_{i=1}^{N}||\boldsymbol{t}_{i1}^* - \boldsymbol{t}_i||_2^2. \tag{22}$$

IV. Experiments

In this section, we present the details of exploited datasets, evaluation metrics, experimental settings, experimental results and model analysis.

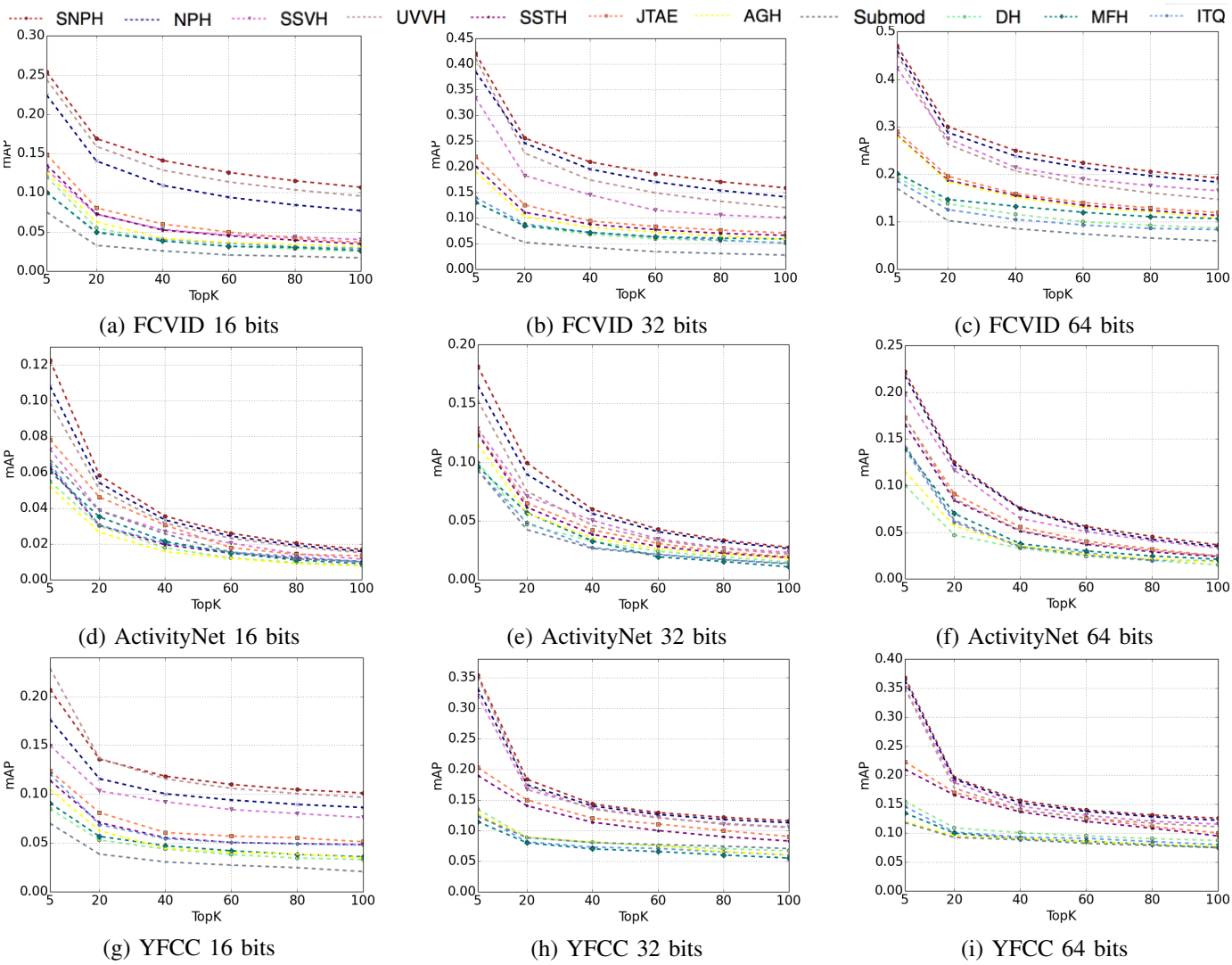
A. Datasets and Evaluation Metrics

To evaluate the effectiveness of our proposed SNPH method, we conducted extensive experiments on three large-scale video collections: FCVID[1], ActivityNet[2] and YFCC[3].

Fudan-Columbia Video Dataset (**FCVID**) [72] consists of 91,223 Youtube videos, which are annotated into 239 categories manually. It covers a wide range of topics such as sports and various events with average length of 167 seconds. We selected 45,585 videos for the train split and the rest 45,600 videos as queries and gallery videos, which was the same with the setting in [32] and [34].

ActivityNet [73] contains around 20K video clips that classified into 200 categories covering a wide range of complicated human daily activities. Particularly, we used 9,722 videos for training split. Since the labels of those videos in original test split were not released, we randomly chose 1000 videos from

[1]https://http://bigvid.fudan.edu.cn/FCVID/
[2]http://activity-net.org/
[3]https://webscope.sandbox.yahoo.com/



(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

Fig. 2: The comparison of search results among a variety of hashing methods in terms of mAP@K over three datasets with various code lengths.

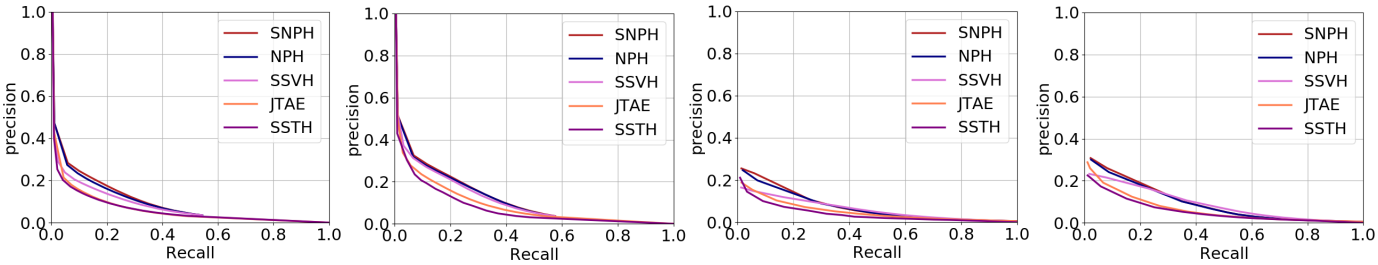
the validation set as query instances, and used the remaining 3,760 validation videos to form the gallery.

Yahoo Flickr Creative Common 100 Million Dataset (**YFCC**) [74] is one of the largest public video dataset, which collects 0.8M video clips with average length of 37 seconds. Due to some damaged data and invalid download links, we employed 511,044 videos for our experiment. In detail, we used 409,788 unlabeled videos for unsupervised training and 101,256 labeled videos for evaluation. There are 80 scenes collected from the third level of MIT SUN scene hierarchy [75] in the test split. Following the setting in [34], we randomly chose 1,000 videos with non-zero label as queries and the remaining labeled videos as search database.

We followed the evaluation protocols in [31], [32], i.e., Hamming ranking, and employed the following two evaluation metrics: 1) We exploited mean Average Precision at top K (mAP@K) retrieved videos as the main evaluation metric to evaluate the performance of hashing methods. mAP@K is defined as the mean of average precision of retrieved relevant videos number in the top K results [76]. 2) We used Precision-Recall (PR) curve as an assisted measurement [77]. Same as previous work [31], [32], we defined ground-truth neighbors for a query if they shared at least one label with this query.

B. Implementation Details

Without loss of generality, we uniformly selected 25 frames to represent each video as [32] and [34] did. For each frame, we used VGG16 [56] pre-trained on ImageNet [78] to extract 4096-D frame features. It is noteworthy that other pre-trained networks such as ResNet [79] also work, and we chose VGG16 for fair comparison with other methods. We used a single attention head for each encoding layer with dimension of the memory state 256. As for K-means clustering, we conducted 15 iterations. We set the size of anchor set n as 2000. We set both numbers of nearest anchors a and e the same as 3. We set the size of pseudo label set q as 600. We set the scaling factor d^k as 256. We empirically set three hyper-parameters $\alpha 1$, $\alpha 2$ and $\alpha 3$ as 0.9, 0.1 and 0.1 respectively. We set the learning



(a) FCVID 32 bits (b) FCVID 64 bits (c) ActivityNet 32 bits (d) ActivityNet 64 bits

Fig. 3: The comparison of precision-recall curves for different video hashing methods on FCVID and ActivityNet with binary codes of different lengths.

rate and mini-batch size as $3 \times 10e^{-4}$ and 256 respectively. We trained our model with Adam optimization algorithm [80]. We set the maximum training epoch as 100 on both FCVID and ActivityNet, and 15 on YFCC. We conducted all these experiments under Pytorch[4] framework on a Geforce GTX 1080 Ti GPU.

Since the derivative of (5) is zero almost anywhere, we chose to use special training expedients as [37] and [81] did. During the forward pass in training, we set $\tau(x) = \mathbf{1}_{\sigma(x)>z},\ z \sim U[0,1]$, where $U[0,1]$ denotes uniform distribution in $[0,1]$ and $\mathbf{1}$ is the indicator function. During back-propagation, we set the derivative of τ as $\frac{\partial \tau}{\partial x}(x) = \sigma(x)(1-\sigma(x))$. During test stage, we calculated τ in a deterministic form with (5). Since the derivative of $sgn()$ in (15) was zero almost everywhere, we referred to BinaryNet [82] to handle the ill-posed gradient problem.

C. Results and Analysis

Comparisons with state-of-the-arts: We used several state-of-the-art unsupervised hashing methods as baselines in the experiment: Iterative Quantization (ITQ) [2], Anchor Graph Hashing (AGH) [68], Submodular video hashing (Submod) [45], Multiple Feature Hashing (MFH) [22], Deep Hashing (DH) [83], Joint Temporal Appearance Encoder (JTAE) [33], Self-Supervised Temporal Hashing (SSTH) [31], Self-Supervised Video Hashing (SSVH), Unsupervised Variational Video Hashing (UVVH) [25], and Neighborhood Preserving Hashing (NPH) [34]. For the consistency of comparison, we kept the experimental settings for all these methods the same.

On FCVID dataset, SNPH consistently outperforms the other compared methods with all code lengths in terms of mAP@K as shown in Fig. 2 (a)-(c). The most competitive methods are NPH and UVVH. Specifically, SNPH outperforms NPH by 2.9%, 2.8%, 3.1% in terms of mAP@5, mAP@20 and mAP@60 respectively with 16 bits. It outperforms UVVH by 3.7% and 4.5%, 2.8% in terms of mAP@20 and mAP@60 respectively with 64 bits. In addition, it outperforms NPH by 3.5% in terms of mAP@5 with 32 bits and 1.4% in terms of mAP@20 with 64 bits. Besides, it outperforms the other baselines by a large margin. We also compared SNPH with two other methods, LA-CODE [35] and DHTA [18]. We listed the mAP@20 results of SNPH, LA-CODE and DHTA with 128 bits in TABLE I. It shows that SNPH significantly outperforms these two methods. Furthermore, the PR curves shown in Fig. 3(a) and (b) indicate that SNPH consistently achieves higher precision than NPH, JTAE, SSTH and SSVH at the same rate of recall.

TABLE I: mAP@20 results of SNPH, LA-CODE and DHTA with 128 bits on FCVID

Methods	SNPH	LA-CODE	DHTA
mAP@20	**0.341**	0.328	0.320

On ActivityNet dataset, SNPH consistently outperforms these state-of-the-art methods with all code lengths in terms of mAP@K as shown in Fig. 2 (d)-(f). Specifically, SNPH outperforms the most competitive NPH by 1.4% and 1.7% in terms of mAP@5 with 16 bits and 32 bits respectively. It should be noticed that the search gallery only contains around 3,000 samples, and there is probably no ground-truth neighbor for some queries. Thus the mAP@K results on ActivityNet for all methods seem not so good as that on FCVID especially with a large K. In addition, PR curves shown in Fig. 3(c) and (d) indicate that SNPH consistently achieves higher precision than the other methods at the same rate of recall.

On YFCC dataset, SNPH outperforms all the other methods with different code lengths except for UVVH as shown in Fig. 2 (g)-(i). Specifically, SNPH significantly outperforms UVVH with 32 bits and 64 bits. While SNPH is not as good as UVVH in view of mAP@5 with 16 bits, it outperforms UVVH when K becomes larger, which shows the strength of SNPH in general. In addition, SNPH outperforms NPH by 3.1%, 2.0%, 1.8% in terms of mAP@5, mAP@20 and mAP@40 with 16 bits. And it outperforms NPH by 2.3% and 1.0% in terms of mAP@5 with 32 bits and 64 bits respectively.

Cross-dataset evaluation comparisons: To investigate how SNPH generalizes to cross-dataset search tasks, we trained various methods on FCVID and test on YFCC and compare the performances. TABLE II lists the cross-dataset performances of various hashing methods in view of mAP@20 with 64 bits. As can be seen, the search performances of all these methods decrease in different degrees compared with training and testing both on YFCC dataset. It indicates that the performance is associated with the scale and domain of the training dataset.

[4]http://pytorch.org/

TABLE II: 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	SNPH
mAP@20	0.155↓ 6.3	0.173↓ 7.8	0.180↓ 6.0	0.183↓ 5.2

TABLE III: mAP@K results of different on FCVID with 64-bit codes.

Methods	K=20	K=40	K=60	K=80	K=100
SNPH-randi	0.289	0.238	0.213	0.196	0.182
SNPH-rnn	0.291	0.241	0.215	0.198	0.184
SNPH-uni	0.292	0.242	0.215	0.198	0.184
SNPH-sin	0.294	0.240	0.213	0.196	0.183
SNPH	**0.300**	**0.250**	**0.224**	**0.206**	**0.192**

SNPH still outperforms the other methods in the cross-dataset evaluation setting. Besides, the performance drop of SNPH is less than the other methods, which indicates its better generalization cross different datasets.

Effectiveness of components of SNPH: Firstly, we evaluated the effect of the neighborhood attention mechanism. We compared SNPH with the following two baselines: 1) SNPH-randi. we randomly initialized the memory states without using neighborhood representation $\boldsymbol{n}_i$, i.e. our proposed neighborhood attention mechanism degraded to conventional self attention mechanism. 2) SNPH-rnn. We substituted m^t with directly the input $\boldsymbol{x}^t$ in (12), i.e. we removed the attention mechanism. All the other settings of these three methods were kept the same. We listed the mAP@K scores of SNPH-randi, SNPH-rnn and SNPH in Table III. By comparing SNPH-randi and SNPH-rnn we can see that the self-attention mechanism itself has little help to learn effective hash functions. This is because it merely refers to the inherent content in a video while ignoring the neighborhood relationships among videos, which is unsuitable for nearest neighbor search. In addition, SNPH consistently outperforms SNPH-randi and SNPH-rnn, which indicates that our proposed neighborhood attention mechanism has significant impact on the final performance. It shows that exploiting the neighborhood relationships properly is significant with regard to the nearest neighbor search task, which is the major contribution of our proposed neighborhood attention mechanism over existing self-attention mechanism.

Next, we tested the effectiveness of the hierarchical structure-adaptive architecture. We compared SNPH with following two baselines. a) SNPH-sin. We substituted the 2-layer LSTM with a single layer LSTM. b) SNPH-uni. We removed the boundary detector, instead we equally divided each video into fixed-length segments and encoded them via a hierarchical recurrent encoder. These two baselines were both equipped with neighborhood mechanisms. We reported the mAP@K results of SNPH-sin, SNPH-uni and SNPH on FCVID with 64 bits in TABLE III. As can be seen, SNPH outperforms these two baselines remarkably. This indicates that the hierarchical recurrent architecture of structure-adaptive encoder is more powerful than a single layer LSTM architecture. Besides, the boundary detector, which better exploits the hierarchical structure of the video, can greatly improve the search accuracy.

TABLE IV: mAP@K results of SNPH with different loss terms on FCVID.

Methods	K=20	K=40	K=60	K=80	K=100
SNPHoL_{vr}	0.256	0.196	0.167	0.147	0.133
SNPHoL_{ar}	0.217	0.179	0.161	0.150	0.142
SNPHoL_{nr}	0.130	0.111	0.102	0.094	0.088
SNPH-L_{vr}	0.224	0.181	0.164	0.153	0.143
SNPH-L_{ar}	0.263	0.202	0.172	0.151	0.138
SNPH-L_{nr}	0.285	0.234	0.209	0.192	0.179
SNPH	**0.300**	**0.250**	**0.224**	**0.206**	**0.192**

TABLE V: mAP@20 results for SNPHf and SNPH on FCVID with various code lengths.

Methods	16 bits	32 bits	64 bits
SNPHf	0.159	0.247	0.290
SNPH	**0.169**	**0.258**	**0.300**

Then we evaluated the impact of each loss term in (16). We denoted SNPH that was trained with only visual content reconstruction loss L_{vr} as SNPHoL_{vr} and trained without L_{vr} as SNPH-L_{vr}. Similarly, we proposed four other baselines SNPHoL_{ar}, SNPH-L_{ar}, SNPHoL_{nr}, SNPH-L_{nr}. We reported the mAP@K results of these baselines on FCVID with 64 bits in TABLE IV. Firstly, we can observe that mAP@K results of SNPHoL_{vr} are better than SNPHoL_{ar} when K is small, but worse than SNPHoL_{ar} when K becomes larger. One possible reason is that the visual content reconstruction loss encourages the model to capture inherent information in the video, thus it is beneficial to search a small number of videos that are most similar. In contrast, when more videos are expected to be searched, exploiting the neighborhood relationships between videos becomes more important. Furthermore, when we remove one of these three losses to train SNPH, the performance drops in different degrees. It indicates that all these loss terms are beneficial to enhance the search accuracy.

To show the advantage of the center reconstruction loss L_{nr} over the neighborhood information reconstruction loss used in the previous work [34], we compared SNPH with SNPHf where L_{nr} was substituted with the original neighborhood reconstruction loss. We reported the mAP@20 results of SNPH and SNPHf on FCVID in TABLE V. It shows that the center reconstruction loss which directly aligns binary vectors to pseudo labels efficiently leads to more discriminative hash functions and further improves the search performance.

TABLE VI: mAP@K results of SNPH-plain, SNPH-rd and SNPH with 64 bits on FCVID.

Methods	K=20	K=40	K=60	K=80	K=100
SNPH-plain	0.260	0.200	0.171	0.151	0.139
SNPH-rd	0.287	0.238	0.212	0.195	0.182
SNPH	**0.300**	**0.250**	**0.224**	**0.206**	**0.192**

We evaluated the anchor set $\{\boldsymbol{u}_i^*\}_{i=1}^n$ and pseudo label set $\{\boldsymbol{t}_i^*\}_{i=1}^q$ built in this work. Accordingly, we proposed two compared baselines. 1) We used two random sets, where all the points were randomly generated, to replace $\{\boldsymbol{u}_i^*\}_{i=1}^n$ and $\{\boldsymbol{t}_i^*\}_{i=1}^q$. The neighborhood representation and pseudo

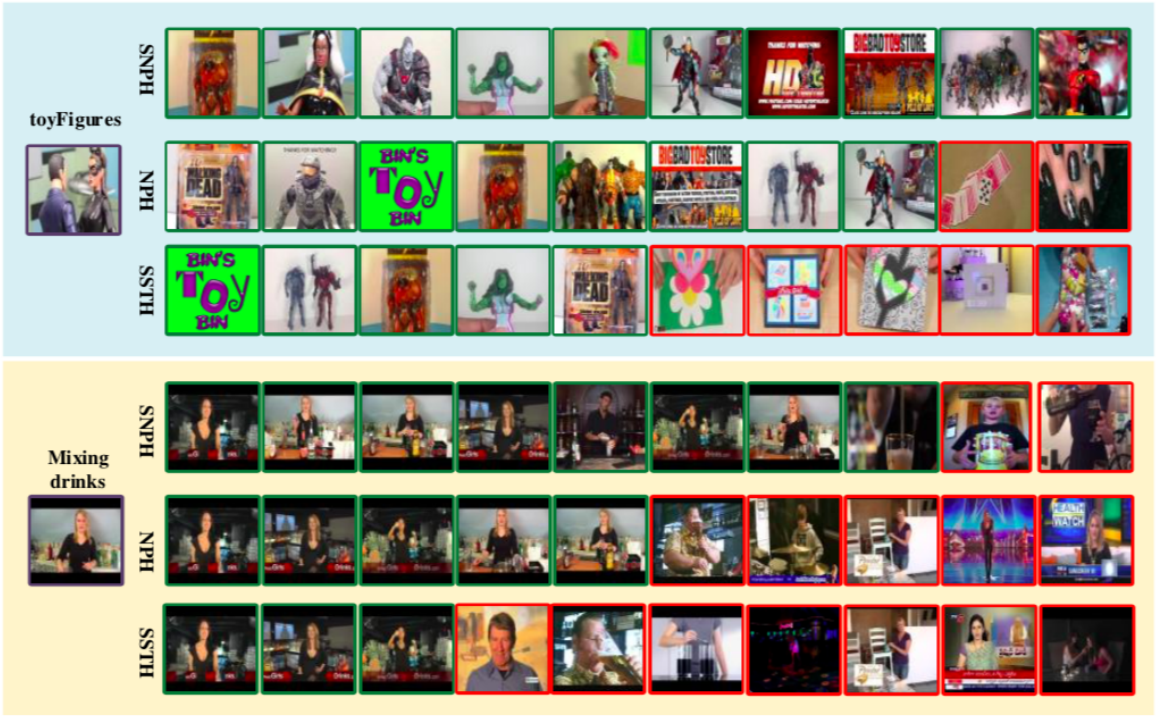


Fig. 4: Top-10 retrieved results. Blue for FCVID and yellow for ActivityNet. Purple border denotes queries. Green borders denote correct retrieved results and red borders denote incorrect retrieved results. According to the qualitative analysis, we can see that our proposed SNPH obtains higher search accuracy than the other methods.

TABLE VII: mAP@K results of SNPH0, SNPH1, SNPH2 and SNPH3 on ActivityNet with 64-bit codes.

Methods	K=5	K=20	K=40	K=60	K=80
SNPH0	0.179	0.100	0.063	0.044	0.035
SNPH1	0.213	0.111	0.068	0.049	0.038
SNPH2	**0.217**	**0.115**	**0.070**	**0.050**	0.039
SNPH3	0.215	0.113	0.069	**0.050**	**0.040**

label were retrieved from these random sets as described in Section III-B and Section III-D. We denoted this baseline as SNPH-rd. 2) We removed the loss term L_{nr} and randomly initialized the memory state, denoting it as SNPH-plain. We listed mAP@K results of SNPH-plain, SNPH-rd and SNPH with 64 bits on FCVID in TABLE VI. We can see that SNPH-rd significantly outperforms SNPH-plain, which indicates that the neighborhood representations and pseudo labels retrieved from a randomly built set can still provide effective guidance. This is because these vectors can reflect the neighborhood relationships among videos, i.e., if two videos are similar, they have similar neighborhood representations and similar pseudo labels. Besides, we can see that SNPH outperforms SNPH-rd. This confirms that a set built by K-means clustering has strong power to represent the vast video data points, and the points in this set can better reflect the neighborhood relationships among videos.

Furthermore, we evaluated the effect of updating the adjacency matrix $\boldsymbol{A}$ to fine-tune the network. Let SNPH0 denote SNPH that is trained without L_{ar} and let $\boldsymbol{A}_0$ denote the adjacency matrix calculated based on outputs of SNPH0. We added L_{ar} which was calculated with $\boldsymbol{A}_0$ to fine-tune SNPH0 and obtain SNPH1. $\boldsymbol{A}_1$ was calculated based on outputs of SNPH1. Similarly, we fine-tuned SNPH1 with guidance of $\boldsymbol{A}_1$ to obtain SNPH2. Likewise, SNPH3 was a fine-tuned version of SNPH2. We presented mAP@K results of SNPH0, SNPH1, SNPH2 and SNPH3 on ActivityNet with 64-bit codes in Table VII. By comparing SNPH1 with SNPH0, we can see that L_{ar} brings great improvement to the search accuracy. This shows that the pairwise similarity derived in the optimized feature space can provide reliable guidance for self-supervised learning. By comparing SNPH1, SNPH2 and SNPH3, we can see that updating the adjacency matrix can bring about further performance improvement, while the improvement becomes less significant with more iterations. Considering both the training efficiency and retrieval accuracy, we can choose SNPH1 as our final version.

Qualitative results: We presented the top-10 retrieved results with 64-bit binary codes of SNPH, NPH and SSTH on FCVID and ActivityNet datasets in Fig. 4. For the sake of brevity, we used a sampled frame to represent the corresponding video. As can be seen, SNPH obtains higher search accuracy than the other methods on both datasets under different backgrounds and shooting angles. For "toyFigure" category, SNPH retrieves all correct videos but NPH retrieves two wrong videos and SSTH is only half right. On ActivityNet, the scene is more complicated, and all these methods make mistakes in different degrees. For example, SNPH mistakes the activity "mixing drinks" with "pouring drinks", while both activities contain the action "pouring". In contrast, NPH and

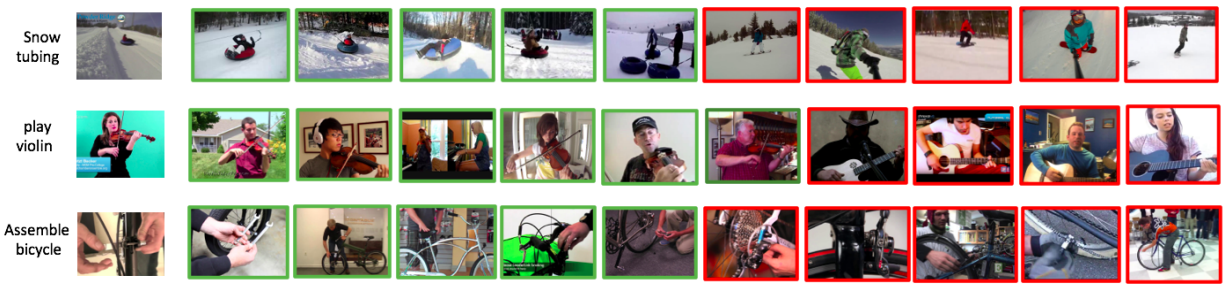


Fig. 5: Some failure cases in top-10 retrieved results of SNPH. Green borders denote correct retrieved results and red borders denote incorrect retrieved results. As can be seen, when two movies contain similar scenes and actions, SNPH tends to consider them to be similar.

Fig. 6: Example boundary detection results on FCVID. Red vertical lines indicate activations of the boundary detector. The rows from top to bottom are videos in categories "making sushi", "gorilla" and "basketball" respectively.

SSTH are more likely to mistake the action pouring with some incorrect actions such as drinking.

We further listed some other failure cases on in Fig. 5. As can be seen, when two movies contain similar scenes and actions, SNPH tends to consider them to be similar. For example, SNPH mistakes "snow boarding" with "snow tubing" since there are snow in the background and people sliding with some vehicles on the snow in these videos. In the second row, SNPH mistakes "play violin" with "play guitar" since violin and guitar look very similar and these activities share similar actions. In the third row, the query video is in category "assemble bicycle", while five wrong videos are in category "fix bicycle". It should be aware that it is hard to distinguish these two activities if no professional label is available. These failure cases may show a limitation of self-supervised learning since videos which contain similar scenes and actions tend to have similar spatio-temporal features via unsupervised learning.

As shown in Fig. 6, we picked several representative boundary detection results on FCVID to observe whether the structure-adaptive encoder well detects the time boundaries across frames in the video. It qualitatively shows that the output of the boundary-aware detector is consistent with our commonsense. The first example is an instruction movie for sushi making. There are abrupt shot changes in this case, and boundaries detected by the encoder are all overlapped with these shot changes. As for the second example, some transitions last for several frames (shown in purple area) and the detected boundaries successfully lay in these areas. However, when there is no such clear shot change as the third example shows, the encoder tends to uniformly divide the video in several short segments. A possible reason is that SNPH tends to shorten the input path such that the content in the long frame sequence can be better captured, which plays a similar role with a conventional hierarchical RNN encoder.

TABLE VIII: Time for encoding, feature extraction frame and Hamming search of different methods with 64-bit codes on FCVID.

Methods	Encoding	Frame feature extraction	Search
SSTH	0.88ms	20.41ms	8.77ms
SSVH	1.03ms		
NPH	1.42ms		
SNPH	1.60ms		

Implementation time: Given a query video, the implementation time includes: 1) The encoding time. Time cost to generate binary codes from a sequence of frame features by using different deep video hashing methods. 2) Frame feature extraction time. Time cost to extract a sequence of frame features via CNN. 3) Hamming search time. Time cost to search nearest neighbor videos for a query video in the Hamming space. It should be aware that the frame feature extraction time and Hamming search time are consistent among different hashing methods. We reported the implementation time of SNPH, NPH, SSTH and SSVH in TABLE VIII on FCVID dataset with 64 bits. We implemented these models in Python, using the Pytorch library. The computing platform was equipped with a 4.0 GHz Intel CPU, 32 GB RAM, and NVIDIA GTX 1080Ti. As can be seen, all these methods are practical for search engines in view of encoding efficiency. As the CNN feature extraction lead to high time cost, we input frame features instead of raw frames to train our model for

sake of training efficiency. Besides, the Hamming search time on a dataset as large as 45,600 is about 8ms, which indicates that our method can scale to large video database.

V. Conclusion

In this paper, we have presented an unsupervised video hash method, structure-adaptive neighborhood preserving hashing for scalable video search. Compared to previous works, SNPH can better preserve neighborhood relationships between videos and make full use of the hierarchical structure of a video. SNPH outperforms state-of-the-arts on three public benchmark datasets. There are several interesting directions for future work. Firstly, the calculation of neighborhood representation is also required during test stage. While the extra time cost is negligible, we can try to avoid such operation by incorporating knowledge distilling into our method. In addition, we directly input extracted CNN features to train our model, which may omit some important information in the raw frames for nearest neighbor search. To further improve the performance, we may use raw frames as input and train the CNN together during the hash model training. Furthermore, we may develop semi-supervised learning methods that use some labels to help the model distinguish hard sample pairs.

References

[1] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *ACM Symposium on the Theory of Computing*, 1998, pp. 604–613.

[2] 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, 2013.

[3] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2009, vol. 282, no. 3, pp. 1753–1760.

[4] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4514–4524, 2016.

[5] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber, "Multimodal similarity-preserving hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 824–830, 2014.

[6] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[7] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[8] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou, "Deep hashing via discrepancy minimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 6838–6847.

[9] L. Duan, J. Lin, Z. Wang, T. Huang, and W. Gao, "Weighted component hashing of binary aggregated descriptors for fast visual search," *IEEE Transactions on Multimedia*, vol. 17, no. 6, pp. 828–842, 2015.

[10] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Context-aware local binary feature learning for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1139–1153, 2018.

[11] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 949–963, Feb 2018.

[12] L. Yuan, T. Wang, X. Zhang, F. E. H. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3080–3089.

[13] Y. Shen, J. Qin, J. Chen, L. Liu, F. Zhu, and Z. Shen, "Embarrassingly simple binary representation learning," in *International Conference on Computer Vision Workshops*, 2019, pp. 2883–2892.

[14] L. Zhu, H. Cui, Z. Cheng, J. Li, and Z. Zhang, "Dual-level semantic transfer deep hashing for efficient social image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2020.

[15] Y. Wang, J. Liang, D. Cao, and Z. Sun, "Local semantic-aware deep hashing with hamming-isometric quantization," *IEEE Transactions Image Processing*, vol. 28, no. 6, pp. 2665–2679, 2019.

[16] H. Cui, L. Zhu, J. Li, Y. Yang, and L. Nie, "Scalable deep hashing for large-scale social image retrieval," *IEEE Transactions Image Processing*, vol. 29, pp. 1271–1284, 2020.

[17] Y. Wang, X. Ou, J. Liang, and Z. Sun, "Deep semantic reconstruction hashing for similarity retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 387–400, 2021.

[18] J. Bai, B. Chen, Y. Li, D. Wu, W. Guo, S. Xia, and E. Yang, "Targeted attack for deep hashing based retrieval," in *European Conference on Computer Vision*, vol. 12346, 2020, pp. 618–634.

[19] G. Ye, D. Liu, J. Wang, and S. Chang, "Large-scale video hashing via structure learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2272–2279.

[20] 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, 2017.

[21] 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.

[22] 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 ACM international conference on Multimedia*, 2011, pp. 423–432.

[23] L. Yu, Z. Huang, J. Cao, and H. T. Shen, "Scalable video event retrieval by visual state binary embedding," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1590–1603, 2016.

[24] V. E. Liong, J. Lu, Y. Tan, and J. Zhou, "Deep video hashing," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1209–1219, 2017.

[25] 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, 2020.

[26] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.

[27] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[28] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, and Q. Dai, "Supervised hash coding with deep neural network for environment perception of intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 284–295, 2018.

[29] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear structural hashing for scalable video search," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1421–1433, 2018.

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

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

[32] 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.

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

[34] S. Li, Z. Chen, J. Lu, X. Li, and J. Zhou, "Neighborhood preserving hashing for scalable video retrieval," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8211–8220.

[35] T. Yu and N. Padoy, "Encode the unseen: Predictive video hashing for scalable mid-stream retrieval," in *Asian Conference on Computer Vision*, vol. 12626, 2020, pp. 427–442.

[36] K. Kim, S. Choi, J. Kim, and B. Zhang, "Multimodal dual attention memory for video story question answering," in *European Conference on Computer Vision*, 2018, pp. 698–713.

[37] L. Baraldi, C. Grana, and R. Cucchiara, "Hierarchical boundary-aware neural encoder for video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3185–3194.

[38] J. Hsieh, S. Yu, and Y. Chen, "Motion-based video retrieval by trajectory matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 396–409, 2006.

[39] D. Zhong and S. Chang, "An integrated approach for content-based video object segmentation and retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1259–1268, 1999.

[40] L. Gao, Z. Li, and A. K. Katsaggelos, "An efficient video indexing and retrieval algorithm using the luminance field trajectory modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 10, pp. 1566–1570, 2009.

[41] Y. Lai and C. Yang, "Video object retrieval by trajectory and appearance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 1026–1037, 2015.

[42] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[43] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2946–2953.

[44] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[45] L. Cao, Z. Li, Y. Mu, and S. Chang, "Submodular video hashing: a unified framework towards video pooling and indexing," in *Proceedings of the ACM Multimedia Conference*, 2012, pp. 299–308.

[46] Y. Li, R. Wang, Z. Huang, S. Shan, and X. Chen, "Face video retrieval with image query via hashing across euclidean space and riemannian manifold," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4758–4767.

[47] S. Qiao, R. Wang, S. Shan, and X. Chen, "Deep heterogeneous hashing for face video retrieval," *IEEE Transactions on Image Processing*, vol. 29, pp. 1299–1312, 2020.

[48] G. Wu, L. Liu, Y. Guo, G. Ding, J. Han, J. Shen, and L. Shao, "Unsupervised deep video hashing with balanced rotation," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 3076–3082.

[49] 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, 2019.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1106–1114.

[51] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," in *Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1494–1504.

[52] Z. Xu, Y. Yang, and A. G. Hauptmann, "A discriminative CNN video representation for event detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1798–1807.

[53] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[54] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[55] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2011.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[57] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4489–4497.

[58] J. Song, Y. Guo, L. Gao, X. Li, A. Hanjalic, and H. T. Shen, "From deterministic to generative: Multi-modal stochastic RNNs for video captioning," *CoRR*, vol. abs/1708.02478, 2017.

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[60] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[61] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: a neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[62] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[63] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence – video to text," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4534–4542.

[64] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1029–1038.

[65] B. Zhao, X. Li, and X. Lu, "HSA-RNN: hierarchical structure-adaptive RNN for video summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7405–7414.

[66] A. Santoro, R. Faulkner, D. Raposo, J. W. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. P. Lillicrap, "Relational recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7310–7321.

[67] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[68] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *International Conference on Machine Learning*, 2011, pp. 1–8.

[69] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *Advances in Neural Information Processing Systems*, 2020.

[70] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *European Conference on Computer Vision*, vol. 11218, 2018, pp. 139–156.

[71] Z. D. Guo, B. Á. Pires, B. Piot, J. Grill, F. Altché, R. Munos, and M. G. Azar, "Bootstrap latent-predictive representations for multitask reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, vol. 119, 2020, pp. 3875–3886.

[72] Y. Jiang, Z. Wu, J. Wang, X. Xue, and S. 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, 2018.

[73] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. 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.

[74] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li, "The new data and new challenges in multimedia research," *CoRR*, vol. abs/1503.01817, 2015.

[75] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.

[76] P. Over, G. Awad, J. G. Fiscus, B. Antonishek, M. Michel, W. Kraaij, A. F. Smeaton, and G. Quénot, "TRECVID 2010 - an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *TREC Video Retrieval Evaluation*, 2014.

[77] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Machine Learning, Proceedings of the Twenty-Third International Conference*, vol. 148, 2006, pp. 233–240.

[78] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[79] 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.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[81] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013.

[82] M. Courbariaux and Y. Bengio, "Binarynet: training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.
[83] V. E. 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.

Shuyan Li received the B.S. degree in electronic information engineering from China University of Geosciences, Wuhan, China, in 2016. She is currently pursuing the Ph.D. degree at the Department of Automation, Tsinghua University. Her current research interests include large scale visual search, video representation learning and hashing learning.

Xiu Li received the Ph.D. degree in computer integrated manufacturing from the Nanjing University of Aeronautics and Astronautics in 2000. Since then, she has been with Tsinghua University, Beijing, China. Her research interests include intelligent system, pattern recognition, and data mining.

Jiwen Lu (M'11-SM'15) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in electrical engineering from the Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2012. He is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include computer vision, pattern recognition, and machine learning. He has authored/co-authored over 200 scientific papers in these areas, where 70+ of them are IEEE Transactions papers and 50+ of them are CVPR/ICCV/ECCV papers. He serves the Co-Editor-of-Chief of the Pattern Recognition Letters, an Associate Editor of the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Biometrics, Behavior, and Identity Science, and Pattern Recognition. He is a member of the Multimedia Signal Processing Technical Committee and the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society, and a member of the Multimedia Systems and Applications Technical Committee and the Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems Society. He was a recipient of the National 1000 Young Talents Program of China in 2015, and the National Science Fund of China for Excellent Young Scholars in 2018, respectively. He is a senior member of the IEEE.

Jie Zhou (M'01-SM'04) received the BS and MS degrees both from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the PhD degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995. From then to 1997, he served as a postdoctoral fellow in the Department of Automation, Tsinghua University, Beijing, China. Since 2003, he has been a full professor in the Department of Automation, Tsinghua University. His research interests include computer vision, pattern recognition, and image processing. In recent years, he has authored more than 100 papers in peer-reviewed journals and conferences. Among them, more than 30 papers have been published in top journals and conferences such as the IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Image Processing, and CVPR. He is an associate editor for the IEEE Transactions on Pattern Analysis and Machine Intelligence and two other journals. He received the National Outstanding Youth Foundation of China Award. He is a senior member of the IEEE.