Neighborhood Preserving Hashing for Scalable Video Retrieval

Shuyan Li[1,4], Zhixiang Chen[1,2,3], Jiwen Lu[1,2,3,*], Xiu Li[1,4], Jie Zhou[1,2,3]
[1]Department of Automation, Tsinghua University, China
[2]State Key Lab of Intelligent Technologies and Systems, China
[3]Beijing National Research Center for Information Science and Technology, China
[4] Graduate School at Shenzhen, Tsinghua University, China
li-sy16@mails.tsinghua.edu.cn; zxchen@mail.tsinghua.edu.cn;
lujiwen@tsinghua.edu.cn; xiu.li@sz.tsinghua.edu.cn; jzhou@tsinghua.edu.cn

Abstract

In this paper, we propose a Neighborhood Preserving Hashing (NPH) method for scalable video retrieval in an unsupervised manner. Unlike most existing deep video hashing methods which indiscriminately compress an entire video into a binary code, we embed the spatial-temporal neighborhood information into the encoding network such that the neighborhood-relevant visual content of a video can be preferentially encoded into a binary code under the guidance of the neighborhood information. Specifically, we propose a neighborhood attention mechanism which focuses on partial useful content of each input frame conditioned on the neighborhood information. We then integrate the neighborhood attention mechanism into an RNN-based reconstruction scheme to encourage the binary codes to capture the spatial-temporal structure in a video which is consistent with that in the neighborhood. As a consequence, the learned hashing functions can map similar videos to similar binary codes. Extensive experiments on three widely-used benchmark datasets validate the effectiveness of our proposed approach.

1. Introduction

Scalable video retrieval seeks similar videos from a large database given a query video. Usually, videos are represented by sampled frames and each frame is characterized by a representative feature. The set of frame features are utilized to identify relevant videos or nearest neighbors. In the face of high dimensional features and large scale datasets, hashing methods have attracted a lot of attention in scalable visual retrieval [1–10]. Video hashing methods encode frame features of each video into a compact binary code while enabling the similarity between videos to be preserved in the Hamming space [11–22]. Among them, learning-based video hashing methods which learn data-dependent and task-specific hashing functions have achieved good search accuracy [23–25].

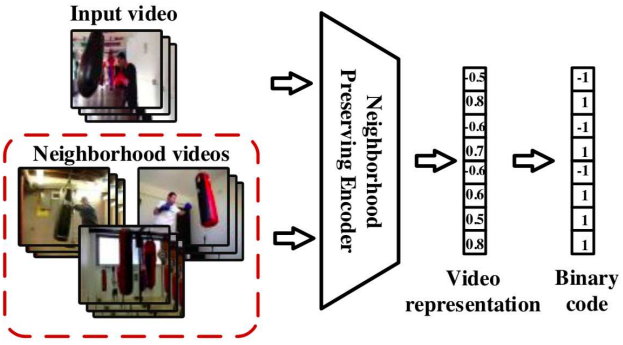


Figure 1: The basic idea of neighborhood preserving hashing. The spatial-temporal neighborhood information is embedded into the encoding network to guide the encoder to preferentially compress relevant visual content of the video into a binary code.

Over the past decade, hashing functions have been integrated into various deep learning architectures to obtain promising performance. Typically, deep video hashing methods compress an entire video into a binary code via deep neural networks and apply similarity preserving criteria on top of the hash layer to learn hashing functions [13, 18–20]. However, since videos contain complex, redundant and sometimes ambiguous content for nearest neighbor search, indiscriminately extracting entire content from a video will inevitably lead to suboptimal hashing functions [26]. On the other hand, manual labels are time and labor consuming especially for large-scale video dataset, which makes supervised hashing methods less feasible for scalable video retrieval.

*Corresponding author

In this work, we propose a Neighborhood Preserving Hashing (NPH) method for scalable video retrieval in an unsupervised manner. As shown in Figure 1, we embed the spatial-temporal neighborhood information of each input video into the NPH encoding network, such that the encoder learns to compress neighborhood-relevant content of the video into a binary code under the guidance of neighborhood information. Specifically, we develop a neighborhood attention mechanism which concentrates on partial useful content in each input frame instead of treating the entire input equally, guided by the neighborhood information. Moreover, we integrate the neighborhood attention mechanism into an RNN-based reconstruction scheme such that the binary codes are encouraged to capture the neighborhood-relevant spatial-temporal structure in a video. Experiments on three widely-used video datasets demonstrate the superior performance of NPH over state-of-the-art methods and also validate the effectiveness of our proposed neighborhood attention mechanism.

2. Related Work

In general, learning-based video hashing methods are classified into supervised methods and unsupervised ones [24, 25].

Supervised learning paradigms are proposed to learn semantically relevant hashing functions by using manually labeled data [11, 13, 15, 27]. For instance, Ye *et al.* [11] proposed Video Hashing with both Discriminative commonality and Temporal consistency (VHDT) to exploit the consistency among successive frames. Liong *et al.* [13] proposed Deep Video Hashing (DVH) which minimized the intra-class variation and maximized the inter-class variation of binary codes to make them discriminative. Yu *et al.* [27] defined a novel metric to select keyframes and applied pairwise constraints to capture the local properties of the events at the semantic level. Their selection rule is hand-crafted and in frame-level, thus is different from ours. In general, supervised methods have achieved overall better performance than unsupervised ones, however, the time and labor consuming labeling requirement makes them impractical for scalable video retrieval.

Unsupervised video hashing methods integrate data properties to learn hashing functions such that the similarity structure between videos is preserved in the Hamming space [12, 16–19, 28]. Early works such as Multiple Feature Hashing (MFH) [12] extended image hashing techniques to video hashing while they ignored to exploit the temporal structure. In recent years, RNNs, which are famous for capturing long-term dependences in a sequence, have been widely used to capture temporal structure in a video [16–18]. Among them, Zhang *et al.* [16] proposed Self-Supervised Temporal Hashing (SSTH) based on a binary autoencoder, which was regarded as the pioneering work in deep unsupervised video hashing. Li. *et al.* [17] extended SSTH by further exploiting the appearance structure. However, both of them neglect to exploit the neighborhood structure. Song *et al.* [18] attempted to remedy this disadvantage by designing a neighborhood similarity loss on top of the hash layer. Wu *et al.* [19] proposed Unsupervised Deep Video Hashing (UDVH) which attempted to balance the variation of each dimension when binarizing video features. They then extended UDVH [28] by replacing the baseline model LSTM [29, 30] with Temporal Segment Networks (TSNs) [31]. Whereas all these methods indiscriminately extract entire content from the video without discriminating whether it is relevant to that in the neighborhood, which unavoidably results in suboptimal hash codes.

3. Approach

3.1. Neighborhood Preserving Encoding

Let $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ denote a set of N videos. For each video, we uniformly sample M frames and process each of them with a conventional Convolution Neural Network (CNN) to gain a set of frame features $\{\boldsymbol{v}_i^m\}_{m=1}^{M} \in \mathbb{R}^{M\times l}$. $\boldsymbol{v}_i^m$ denotes the m-th frame feature of the i-th video in $\mathbb{R}^l$. We aim to learn a nonlinear mapping to transfer frame features $\{\boldsymbol{v}_i^m\}_{m=1}^{M}$ to a k-bit binary code $\boldsymbol{b}_i \in \{-1, 1\}^k$ such that the similarity structure between videos is well preserved in the Hamming space.

Unlike most existing deep video hashing methods which indiscriminately extract entire content from the video, we preferentially encode the neighborhood-relevant content of the video into a binary code, guided by the neighborhood information. Specifically, we embed spatial-temporal neighborhood information into the encoding network via a proposed neighborhood attention mechanism. Since the neighborhood information of similar videos tends to be similar, it can guide to project similar videos to similar binary codes. The encoder first maps input frame features $\{\boldsymbol{v}_i^m\}_{m=1}^{M}$ to a neighborhood preserving video representation $\boldsymbol{t}_i$ under the guidance of the spatial-temporal neighborhood information:

$$\boldsymbol{t}_i = \mathcal{E}(\{\boldsymbol{v}_i^m\}_{m=1}^{M}, \boldsymbol{n}_i, \theta). \tag{1}$$

where $\boldsymbol{n}_i$ is the spatial-temporal neighborhood representation in $\mathbb{R}^b$, θ is the learnable parameter set of the encoding network, and $\mathcal{E}$ is a nonlinear projection. Then it discretizes $\boldsymbol{t}_i$ into the neighborhood preserving binary code $\boldsymbol{b}_i$:

$$\boldsymbol{b}_i = sgn(\boldsymbol{t}_i), \tag{2}$$

where $sgn(x) = 1$ if $x \geq 0$ and $sgn(x) = -1$ otherwise.

Neighborhood attention mechanism: The neighborhood attention mechanism is to preferentially incorporate content of an input frame conditioned on the spatial-temporal neighborhood information, which is inspired by

relational recurrent neural networks (rRNN) [32]. The core of rRNN is a memory state which updates by attending over the former memory state and the new input. At time-step t, a new input frame feature $\boldsymbol{v}_i^t$ is encoded into the memory state $m_{i,t}$ as below:

$$m_{i,t} = \text{softmax}\left(\frac{m_{i,t-1}W^q([m_{i,t-1};\boldsymbol{v}_i^t]W^k)^T}{\sqrt{d^k}}\right) \times [m_{i,t-1};\boldsymbol{v}_i^t]W^v, \quad (3)$$

where $m_{i,t}$ is in $\mathbb{R}^b$ and $m_{i,0}$ is a random initialized vector. d^k is a scaling factor. $[x_1;x_2]$ means the row-wise concatenation of x_1 and x_2. W^x denotes a learnable parameter matrix. (3) is an extension of self-attention where $[m_{i,t};\boldsymbol{v}_i^t]$ is used to calculate the key and value, and $m_{i,t}$ is used to calculate the query [33]. It forms the attention over the memory state and the new input. This mechanism is able to learn to determine which content in input frame should be preferentially written into the memory state conditioned on what is contained in the memory state.

While rRNN is capable to exploit the spatial-temporal structure in a video, merely referring to the inherent content in a video will make the learned hashing functions less effective for scalable retrieval. In order to exploit the neighborhood-relevant spatial-temporal structure, we formulate neighborhood attention mechanism via embedding the spatial-temporal neighborhood information into the memory state. We assume that for each input video $\boldsymbol{x}_i$, a representation $\boldsymbol{n}_i$ has been well developed to carry the neighborhood information. We integrate $\boldsymbol{n}_i$ into the memory state such that the spatial-temporal neighborhood information guides to concentrate attention on relevant content in the input frame. Among a variety of ways to incorporate $\boldsymbol{n}_i$ into the memory state, we choose to inject it only at the first time-step in case of diluting the content from the input video as below:

$$m_{i,1} = \text{softmax}\left(\frac{\boldsymbol{n}_i W^q([\boldsymbol{n}_i;\boldsymbol{v}_i^1]W^k)^T}{\sqrt{d^k}}\right)[\boldsymbol{n}_i;\boldsymbol{v}_i^1]W^v. \quad (4)$$

When $t > 1$, the memory state $m_{i,t}$ updates with (3). Since the neighborhood information has been embedded into the memory state, at each time-step, it will interact with the new input frame and guide to incorporate neighborhood-relevant content into the memory state.

We integrate the neighborhood attention mechanism into a standard LSTM network to form the neighborhood preserving encoding network. We design the gates and cell updates as follows:

$$i_{i,t} = \sigma(W^{iv}\boldsymbol{v}_i^t + W^{ih}h_{i,t-1}) \quad (5)$$

$$f_{i,t} = \sigma(W^{fv}\boldsymbol{v}_i^t + W^{fh}h_{i,t-1}) \quad (6)$$

$$o_{i,t} = \sigma(W^{ov}\boldsymbol{v}_i^t + W^{oh}h_{i,t-1}) \quad (7)$$

$$c_{i,t} = BN(f_{i,t} \odot c_{i,t-1} + i_t \odot \text{MLP}(m_{i,t})) \quad (8)$$

$$h_{i,t} = o_{i,t} \odot tanh(c_{i,t}), \quad (9)$$

where MLP denotes multiple layers perceptron and BN denotes batch normalization. σ denotes sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$. $tanh$ denotes hyperbolic tangent function: $tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. It is worth to notice that the term $m_{i,t}$ takes place of the input vector $\boldsymbol{v}_i^t$ in a standard LSTM. During the encoding period, the neighborhood attention mechanism is considered to suppress the irrelevant or even misleading content in the input frame. We map the hidden state vector of the last time-step $h_{i,M}$ to the video representation $\boldsymbol{t}_i$ via a fully connected (FC) layer:

$$\boldsymbol{t}_i = FC(h_{i,M}, k), \quad (10)$$

where $FC(x,y)$ denotes a linear function that maps the vector x to a vector in $\mathbb{R}^y$. In this way, the neighborhood preserving encoder can learn to capture useful spatial-temporal structure in the video conditioned on the neighborhood information, thus better preserve the neighborhood structure.

Neighborhood representation calculation: The neighborhood representation $\boldsymbol{n}_i$ contains the spatial-temporal neighborhood information of the i-th video. We formulate it as an integration of the spatial-temporal representations of nearest neighbors of the input video.

For each video $\boldsymbol{x}_i$ in the database, we employ an LSTM autoencoder [34] to acquire a spatial-temporal feature $\boldsymbol{y}_i$ in $\mathbb{R}^d$ from the last hidden state of the LSTM encoder. We use $\{\boldsymbol{y}_i\}_{i=1}^N$ to denote the spatial-temporal feature set of training videos. Then we perform K-means clustering on $\{\boldsymbol{y}_i\}_{i=1}^N$ to obtain n centers $\{\boldsymbol{u}_j^*\}_{j=1}^n$ in $\mathbb{R}^{n\times d}$. We consider $\{\boldsymbol{u}_j^*\}_{j=1}^n$ to be an anchor set. We treat the construction of $\{\boldsymbol{u}_j^*\}_{j=1}^n$ as a preprocessing step, thus we will not spend extra time on this step for future training and evaluation. Moreover, we set $n \ll N$, thus the anchor set does not require much storage space.

For the i-th input video, we retrieve a nearest anchors by ranking the distances between $\boldsymbol{y}_i$ and all the anchors in $\{\boldsymbol{u}_j^*\}_{j=1}^n$. We employ l_2 norm for the distance calculation. We set $a \ll n \ll N$, thus calculating a nearest anchors requires little extra time. We use $\boldsymbol{u}_{i1}^*, \boldsymbol{u}_{i2}^*, ..., \boldsymbol{u}_{ia}^*(i1, i2, ..., ia \in \{1, 2, ..., n\})$ to denote the a nearest anchors for the i-th input video and concatenate them in a row-wise fashion. We then project the concatenation via an FC layer to obtain the spatial-temporal neighborhood representation $\boldsymbol{n}_i$ as follows:

$$\boldsymbol{n}_i = FC([\boldsymbol{u}_{i1}^*;\boldsymbol{u}_{i2}^*;..;\boldsymbol{u}_{ia}^*], b). \quad (11)$$

3.2. Neighborhood Preserving Learning

Inspired by the success of reconstruction in hashing function learning [16–18,35,36], we develop an RNN-based reconstruction scheme as illustrated in Figure 2. Unlike most existing methods which only reconstruct the input features, we design several reconstruction principles and cor-

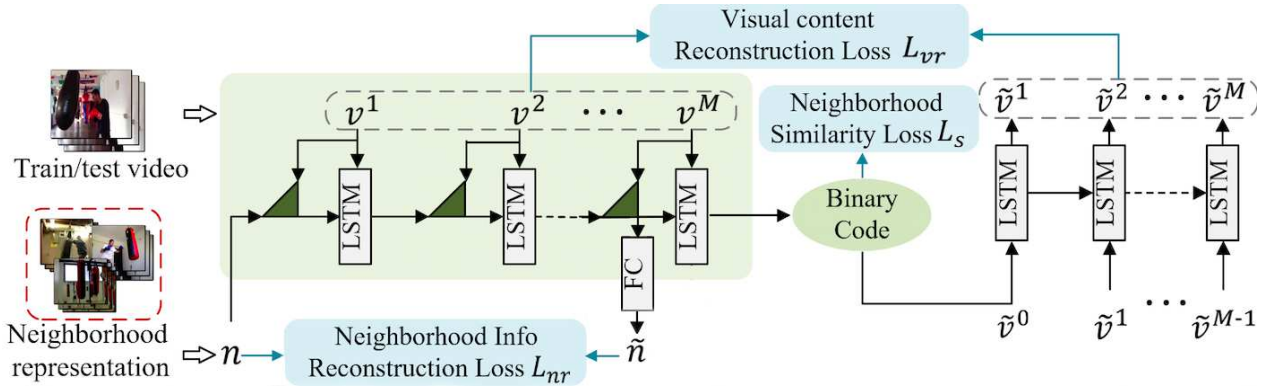


Figure 2: RNN-based reconstruction pipeline. Green area denotes the encoding period. Triangle denotes the neighborhood attention mechanism. Binarization, neighborhood representation calculation, and frame feature extraction are omitted.

responding losses to learn neighborhood preserving hashing functions. 1) We reconstruct the neighborhood structure from the binary codes to ensure that the neighborhood structure is preserved in the Hamming space. Accordingly, we design a neighborhood similarity loss L_s to describe the discrepancy between the neighborhood structure in spatial-temporal feature space and that in the Hamming space. 2) To ensure the binary code to contain visual content in the video, we design an RNN decoder to reconstruct the frame features. We use visual content reconstruction loss L_{vr} to describe the discrepancy between input frame features and reconstructed ones. 3) The neighborhood information is expected to provide the guide in the entire encoding stage, thus it should be contained in the last memory state $m_{i,M}$. Therefore, we reconstruct the spatial-temporal neighborhood representation from $m_{i,M}$. We design a neighborhood information reconstruction loss L_{nr} to denote the discrepancy between the spatial-temporal neighborhood representation and the reconstructed one. In summary, we design the training loss L as the combination of these three losses:

$$L = \alpha_1 L_s + \alpha_2 L_{vr} + \alpha_3 L_{nr}, \tag{12}$$

where α_1, α_2, and α_3 are hyper-parameters that balance these three losses.

Neighborhood similarity loss. To calculate the neighborhood similarity loss L_s, we need to model the neighborhood structure in the spatial-temporal feature space in advance. Instead of building a kNN graph as [18] did, we choose to build an approximate neighborhood graph $\boldsymbol{S}$ based on a small anchor set for the sake of computation efficiency. Each entry of it $S_{ij} \in \{-1, 1\}$ denotes the similarity between spatial-temporal features of the i-th and the j-th training video, with i and $j \in \{1, 2, ..., N\}$. As described in subsection 3.1, firstly we build a spatial-temporal feature set $\{\boldsymbol{y}_i\}_{i=1}^N$ and an anchor set $\{\boldsymbol{u}_i^*\}_{i=1}^n$. Then for each spatial-temporal feature $\boldsymbol{y}_i$, we calculate its a nearest anchors $\boldsymbol{u}_{i1}^*, \boldsymbol{u}_{i2}^*, ..., \boldsymbol{u}_{ia}^*$. With these variables, we can obtain a truncated similarity matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times n}$. Each entry of it is calculated as:

$$Y_{ij} = \begin{cases} \frac{\exp\left(-Dist(\boldsymbol{y}_i, \boldsymbol{u}_j^*)/t\right)}{\sum_{j'=1}^{a} \exp\left(-Dist(\boldsymbol{y}_i, \boldsymbol{u}_{i,j'}^*)/t\right)}, \forall j \in \langle i \rangle \\ 0, \quad \text{otherwise} \end{cases} \tag{13}$$

where $\langle i \rangle$ denotes the indices of a nearest anchors of $\boldsymbol{y}_i$. $Dist()$ is a distance calculation function and we use l_2 norm. t is a bandwidth parameter. According to [37, 38], an approximate adjacency matrix $\boldsymbol{A}$ is calculated as:

$$\boldsymbol{A} = \boldsymbol{Y} \boldsymbol{\Lambda}^{-1} \boldsymbol{Y}^T, \tag{14}$$

where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{Y}^T \boldsymbol{1}) \in \mathbb{R}^{n \times n}$. The approximate adjacency matrix $\boldsymbol{A}$ is a nonnegative and sparse matrix, where entries of each row or column sum to 1. We set $S_{ij} = 1$ if the (i, j)-th entry of the approximate adjacency matrix $A_{ij} > 0$ and $S_{ij} = -1$ otherwise. We treat the construction of $\boldsymbol{S}$ as a preprocessing step for computation efficiency.

We define the similarity between two binary codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ as $\tilde{S}_{i,j} = \frac{1}{k} \boldsymbol{b}_i^T \boldsymbol{b}_j$. For steady training, we substitute $\tilde{S}_{i,j}$ with an approximate one $\hat{S}_{i,j}$: $\hat{S}_{ij} = \frac{1}{k} \boldsymbol{t}_i^T \boldsymbol{t}_j$, where $\boldsymbol{t}_i$ is the k-D neighborhood preserving representation introduced in subsection 3.1. We use Mean Square Error (MSE) to formulate the discrepancy between the neighborhood structure in spatial-temporal feature space and that in the Hamming space. In addition, we introduce an auxiliary MSE term with regard to $\boldsymbol{b}_i$ and $\boldsymbol{t}_i$ to minimize the discrepancy between $\tilde{S}_{ij}$ and $\hat{S}_{i,j}$. Thus we have the specific form of neighborhood similarity loss L_s for (12):

$$L_s = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (S_{ij} - \frac{1}{k} \boldsymbol{t}_i^T \boldsymbol{t}_j)^2 + \frac{1}{kN} \sum_{i=1}^{N} ||\boldsymbol{b}_i - \boldsymbol{t}_i||_2^2. \tag{15}$$

Visual content reconstruction loss: To ensure that the binary code captures the visual content in the video, we employ an LSTM decoder to reconstruct the frame features

from the binary code, and minimize the discrepancy between input frame features $\{\boldsymbol{v}_i^m\}_{m=1}^M$ and the reconstructed ones $\{\tilde{\boldsymbol{v}}_i^m\}_{m=1}^M$. In detail, we project the binary code $\boldsymbol{b}_i$ to a real-value vector $\tilde{\boldsymbol{v}}_i^0 \in \mathbb{R}^l$. At the first time-step, we input $\tilde{\boldsymbol{v}}_i^0$ into the decoder and obtain the first reconstructed frame feature $\tilde{\boldsymbol{v}}_i^1 \in \mathbb{R}^l$ from the output of the decoder. Then we inject $\tilde{\boldsymbol{v}}_i^1$ into the decoder to obtain $\tilde{\boldsymbol{v}}_i^2$. We conduct similar operations recurrently till $\tilde{\boldsymbol{v}}_i^M$ is yielded. We formulate the visual content reconstruction loss L_{vr} with MSE as:

$$L_{vr} = \frac{1}{lMN} \sum_{i=1}^{N} \sum_{m=1}^{M} ||\boldsymbol{v}_i^m - \tilde{\boldsymbol{v}}_i^m||_2^2. \tag{16}$$

Neighborhood information reconstruction loss: We linearly project the last memory state of the encoder $m_{i,M}$ into a b-D vector $\tilde{\boldsymbol{n}}_i$ via an FC layer: $\tilde{\boldsymbol{n}}_i = FC(m_{i,M}, b)$. Then we minimize the discrepancy between $\boldsymbol{n}_i$ and the reconstructed one $\tilde{\boldsymbol{n}}_i$. We formulate the neighborhood information reconstruction loss L_r with MSE:

$$L_{nr} = \frac{1}{Nb} \sum_{i=1}^{N} ||\boldsymbol{n}_i - \tilde{\boldsymbol{n}}_i||_2^2. \tag{17}$$

4. Experimental Results

4.1. Datasets and Experimental Settings

We conducted experiments on three benchmark datasets: FCVID [39], YFCC [40] and ActivityNet [41]. **FCVID** contains 91,223 web videos annotated manually into 239 categories. The total duration of all videos is 4,232 hours and the average duration per video is 167 seconds. Due to the damaged data and category overlap, we collected 91,185 videos of them. Following the setting in [16], we used 45,585 videos for training and 45,600 videos as queries and retrieval database. **YFCC** is a huge collection of multimedia data which contains 0.8M videos. We collected 700,882 videos of them where 409,788 unlabeled data were used for unsupervised learning. Among the 101,256 labeled data, we randomly chose 1000 videos as queries and the rest as retrieval database. **ActivityNet** comprises 20K videos in 200 activity categories collected from YouTube. The lengths of the videos range from several minutes to half an hour. The total length of the whole dataset is 648 hours. We used 9,722 videos for training. Since the test split was not publicly available, we used the validation set as our test set. We randomly sampled 1000 videos from the validation set as queries and used the remaining 3,760 validation videos as the retrieval database.

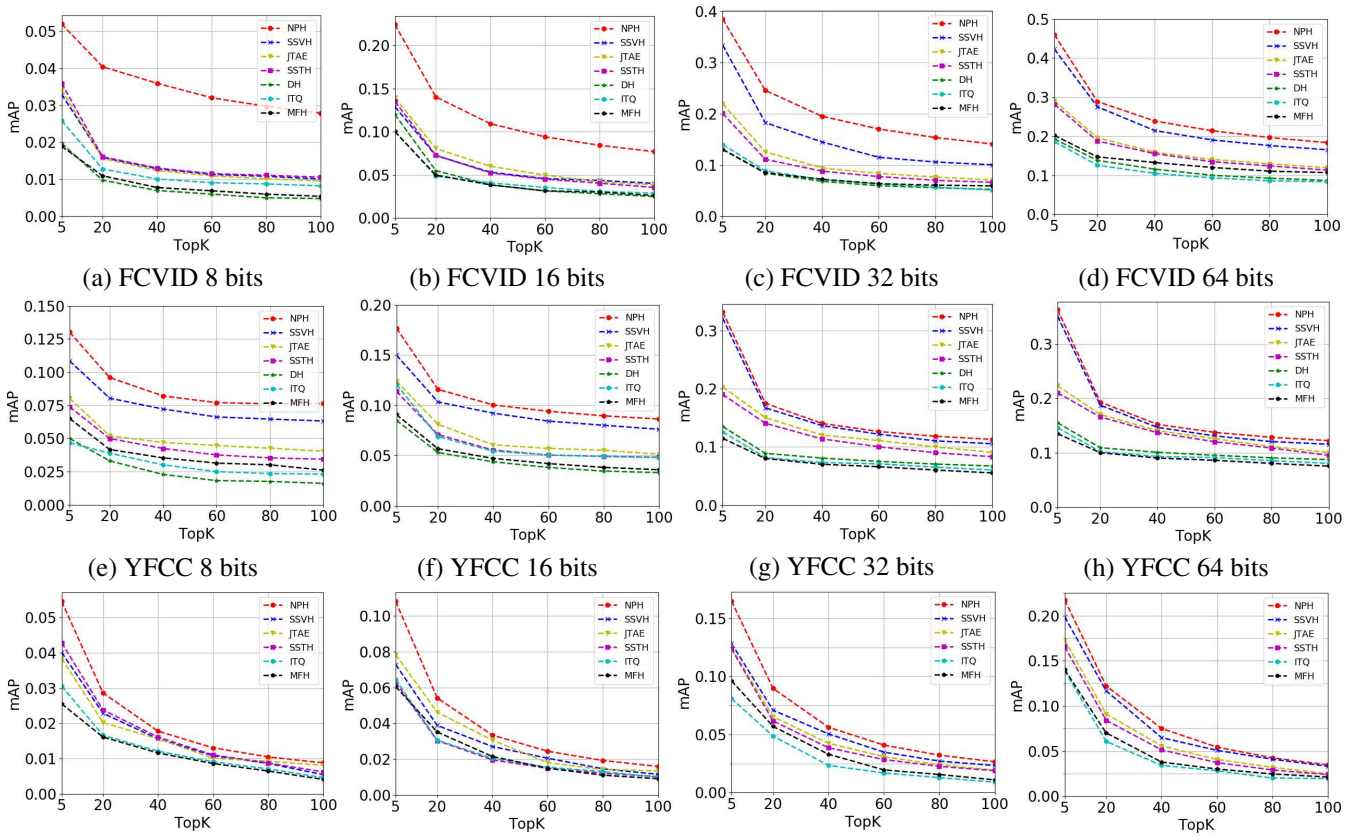
We employed Average Precision at top-K retrieved videos (AP@K) for retrieval performance evaluation [42]. AP@K is defined as AP@K $= \frac{1}{min(R,K)\sum_{i=1}^{K}\frac{R_i}{i}\times I_i}, (1 \leq i \leq K)$. R is the number of total relevant videos in the database. R_i is the number of relevant videos in the top-i retrieved results. $I_i = 1$ if the i-th retrieved video is considered to belong to the same category with the query and $I_i = 0$ otherwise. We defined two samples to be in the same category as long as they shared at least one similar label. We used the mean of AP@K over all the queries (mAP@K) as the main evaluation metric. We used Precision-Recall (PR) curve as an auxiliary evaluation measurement for detailed observation of the retrieval performances. To sort the results, we ranked videos based on Hamming distances from the query video. We chose to evaluate the performances over binary codes with lengths of 8, 16, 32 and 64 bits.

We uniformly sampled 25 frames from each video and used the 16 layers VGG network [43] pre-trained on ImageNet [44] to extract 4096-D frame features. We set both the dimension of the memory state b and the dimension of anchors d as 256. We only used the training videos to obtain the anchor set in subsection 3.1, thus there was no overlap between the anchor set and query/retrieval set. This was to ensure that the encoder attended to the neighborhood-relevant content of the video instead of simply remembering the video ids. We conducted 10 iterations for K-means clustering to obtain an anchor set with 2000 anchors. We set the number of acquired nearest anchors a and the scaling factor d^k as 3 and 256. We empirically set the hyper-parameters α_1, α_2 and α_3 as 0.1, 0.8, and 0.1 respectively to balance those three losses. We applied Drop-Out [45] to avoid overfitting. We initialized parameters of the network with Xavier initialization [46]. We set the learning rate, the momentum and mini-batch size as 0.001, 0.9, and 128 respectively. We trained our model with Adam optimization algorithm [47] and stopped training at the 100th epoch. Since the derivative of $sgn()$ in (2) was 0 almost everywhere, we referred to BinaryNet [48] to handle the ill-posed gradient problem. We conducted all the experiments with Pytorch on single Geforce GTX 1080 Ti GPU.

4.2. Results and Analysis

Comparisons with state-of-the-arts: We compared NPH with the following state-of-the-art unsupervised hashing methods to validate the effectiveness: ITQ [1], DH [2], MFH [12], SSTH [16], JTAE [17] and SSVH [18]. Since ITQ and DH were originally designed for image hashing, we extended them to video hashing as [18] did. The experimental settings for all the methods were the same.

The mAP@K results on FCVID are shown in Figure 3(a)-(d). As can be seen, NPH outperforms MFH, ITQ, DH, SSTH and JTAE by a great margin. Among these methods, ITQ, DH and MFH, which learn the video representation and hashing functions separately, have generally poorer performances than the other compared methods which simultaneously learn the video representation and hashing functions. While SSTH exploits a more delicate stacked BLSTM encoder-decoder structure, NPH outper-



(a) FCVID 8 bits (b) FCVID 16 bits (c) FCVID 32 bits (d) FCVID 64 bits

(e) YFCC 8 bits (f) YFCC 16 bits (g) YFCC 32 bits (h) YFCC 64 bits

(i) ActivityNet 8 bits (j) ActivityNet 16 bits (k) ActivityNet 32 bits (l) ActivityNet 64 bits

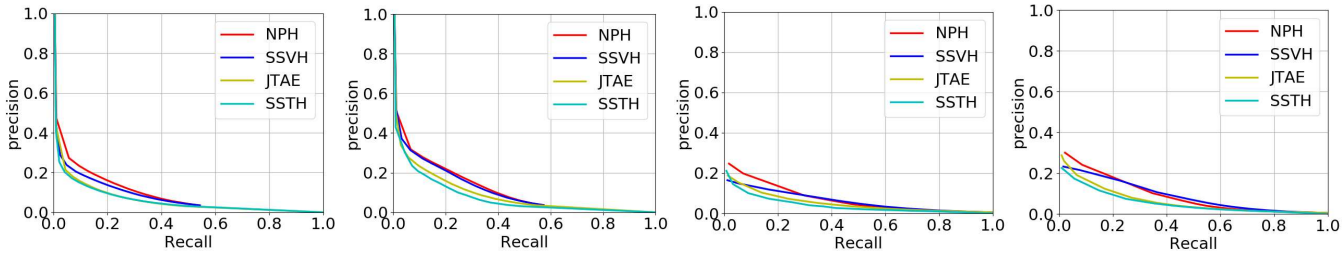
Figure 3: Retrieval performances among all hashing methods in terms of mAP@K over three datasets.

forms it by a large margin with all the code lengths. Besides, while JTAE endeavors to further exploit the appearance structure, NPH outperforms it remarkably. We owe the great advantage of NPH over these two methods to the full use of neighborhood structure. Compared to the most competitive SSVH, NPH consistently shows the superiority. Specifically, NPH outperforms SSVH by around 70% with code length of 16 bits in terms of mAP. Since SSVH also applies a neighborhood similarity loss on top of the hash layer, we owe the superior performance of NPH over SSVH to the neighborhood attention mechanism. It should be noticed that SSVH is built on a hierarchical LSTM structure [49], while NPH is built on a single layer LSTM structure. Since the neighborhood attention mechanism is orthogonal to delicate structures such as hierarchical LSTM, the performance of NPH will be further boosted when more powerful baseline model is employed.

The mAP@K results on YFCC are shown in Figure 3(e)-(h). As can be seen, NPH consistently outperforms the other methods with all code lengths, which validates the effectiveness of it. SSVH is a strong competitor at the code length of 32 and 64 bits. The performance gap between NPH and SSVH becomes marginal compared to that of FCVID. One possible reason is that the scale of YFCC is much larger than that of FCVID but we keep the scale of anchor set the same. A larger anchor set can make the advantage of NPH more prominent on YFCC.

The mAP@K results on ActivityNet are shown in Figure 3(i)-(l). The results of all the methods on ActivityNet are not as good as those on the other two datasets. This may be because many of the videos in this dataset are shot by amateurs in uncontrolled environments, which makes retrieval more difficult. In addition, the scale of the retrieval database is rather small, thus some queries do not have enough true neighbors. Nevertheless, NPH consistently outperforms the state-of-the-art methods with all code lengths, which demonstrates the power of our method.

The PR curves of NPH, SSVH, JTAE, and SSTH are shown in Figure 4. As can be seen, NPH delivers higher precision than state-of-the-art methods at the same rate of recall on FCVID. On ActivityNet, NPH consistently delivers higher precision than JTAE and SSTH. And it has higher precision than SSVH with lower recall required. This is appreciated in approximate nearest neighbor search because



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

Figure 4: PR curves of different video hashing methods with a variety of code lengths on FCVID and ActivityNet.

Table 1: mAP@K results of unseen classes retrieval.

Methods	K=5	K=20	K=40	K=60
SSTH	0.249	0.131	0.080	0.057
JTAE	0.258	0.139	0.086	0.062
SSVH	0.300	0.169	0.102	0.071
NPH	**0.306**	**0.175**	**0.109**	**0.079**

Table 2: mAP@K results when training on train25.

Methods	K=5	K=20	K=40	K=60
SSTH	0.279	0.160	0.098	0.068
JTAE	0.288	0.166	0.103	0.074
SSVH	0.320	0.185	0.110	0.079
NPH	**0.327**	**0.193**	**0.118**	**0.085**

large scale video retrieval is most interested in the high probability of retrieving true neighbors rather than finding out all the neighbors.

Transfer scenario: In order to see how well NPH can be applied to conduct retrieval for unseen classes, we follow [50] to split the dataset into two parts without class overlap: train75 and train25/test25, where train75 is the training set and train25/test25 is the retrieval database/query set. Without loss of generality, we only conduct the evaluation on FCVID. The train25/test25 contains data in 40 classes which are randomly chosen, and the train75 consists of data in the rest classes. Test25 consists of 1000 query videos and train25 consists of the others. The mAP@K results are shown in Table 1. It shows that NPH outperforms the compared methods when retrieving data in unseen classes.

We use train25 for further training and report the mAP@K results in Table 2. It shows that NPH still outperforms state-of-the-art methods.

Ablation study: To evaluate the effectiveness of different components of NPH, we propose these following baselines. FullCNN, Plain and SelfAtt are trained with only the visual content reconstruction loss L_{vr}. The encoding networks of them are fully convolutional networks [51], standard LSTM networks [29, 30] and rRNN [32] respectively. NeibAtt shares the same structure with NPH but is trained with only L_{vr}. SelfAtt+L_s extends SelfAtt by adding the neighborhood similarity loss L_s during training. NeibCat is similar with SelfAtt+L_s except for a little difference in the encoder: a nearest anchors are further concatenated with the encoded video representation, and the concatenation is then mapped into a binary code. NPH- is the same with NPH except that the neighborhood information reconstruction loss L_{nr} is removed.

Table 3: mAP@K results of different methods on FCVID. The rows above are with 32-bit codes and the ones below are with 64-bit codes.

Methods	K=20	K=40	K=60	K=80	K=100
FullCNN	0.189	0.149	0.130	0.117	0.105
Plain	0.175	0.143	0.129	0.119	0.107
SelfAtt	0.187	0.140	0.112	0.103	0.093
NeibAtt	0.201	0.154	0.131	0.117	0.107
SelfAtt+L_s	0.210	0.162	0.138	0.121	0.113
NeibCat	0.214	0.167	0.142	0.124	0.115
NPH-	0.240	0.190	0.166	0.149	0.137
NPH	**0.246**	**0.195**	**0.170**	**0.154**	**0.141**
FullCNN	0.233	0.177	0.150	0.130	0.117
Plain	0.228	0.173	0.146	0.129	0.116
SelfAtt	0.239	0.176	0.146	0.127	0.113
NeibAtt	0.238	0.177	0.147	0.128	0.114
SelfAtt+L_s	0.244	0.203	0.184	0.172	0.162
NeibCat	0.254	0.213	0.193	0.181	0.170
NPH-	0.286	0.238	0.212	0.193	0.179
NPH	**0.294**	**0.240**	**0.213**	**0.196**	**0.183**

We show the mAP@K results of NPH and these baselines with code lengths of 32 bits and 64 bits in Table 3. Remarkably, NPH outperforms these baselines by a great margin. FullCNN, Plain and SelfAtt, which do not take neighborhood structure into consideration, have the poorest performances in general. This shows that exploiting neighborhood structure is beneficial to hashing function learning. Besides, CNN based models do not show significant advantage over RNN based models in our case. NeibAtt

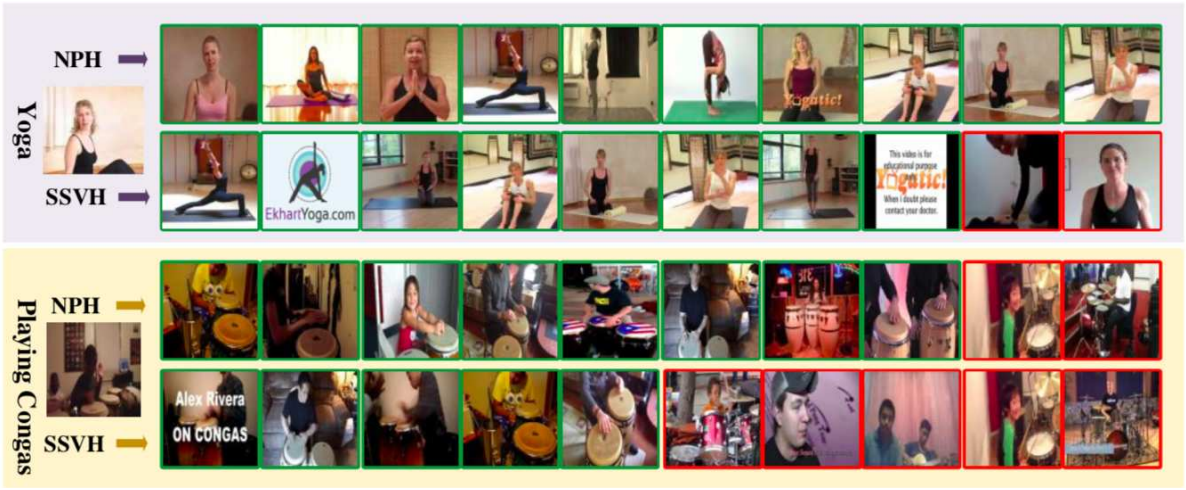


Figure 5: Top-10 retrieved results. Purple for FCVID and yellow for ActivityNet. Rows above are retrieved results of NPH and ones below are that of SSVH. Green border means correct retrieved result and red border means incorrect retrieved result.

outperforms SelfAtt in most cases, which shows the superiority of neighborhood attention mechanism over self-attention mechanism. Specifically, the proposed neighborhood mechanism is able to incorporate the neighborhood-relevant content of the video thus better preserves the neighborhood structure. SelfAtt+L_s shows better performance over NeibAtt, and this indicates that corresponding similarity loss is useful to learn the neighborhood preserving hashing functions. Comparing NPH- and SelfAtt+L_s, we can see that embedding the spatial-temporal neighborhood information into the encoding network can greatly improve the retrieval performance. In addition, that NPH- outperforms NeibConcat indicates that the neighborhood attention mechanism is more than simply assigning anchors to a query. In contrast, it learns what part of each input frame to focus upon conditioned on the spatial-temporal neighborhood information, thus preferentially compresses neighborhood-relevant content into the binary code. Comparing NPH- and NPH, we see that the neighborhood information reconstruction loss further brings improvement.

Qualitative results: We show the top-10 retrieved results with 64 bits of NPH and SSVH on FCVID and ActivityNet datasets in Figure 5. As can be seen, in general, NPH obtains higher retrieval accuracy. For instance, given a query video in category "Yoga", NPH obtains correct top10 retrieved videos while SSVH fails to distinguish this action from similar ones such as "Making shorts". Besides, NPH is able to retrieve relevant videos with various backgrounds and shooting angles.

5. Conclusion

In this paper, we propose NPH for unsupervised scalable video retrieval, which embeds the spatial-temporal neighborhood information into the encoding network so that the neighborhood-relevant content in a video can be compressed in the binary code. Specifically, we propose a neighborhood attention mechanism to preferentially incorporate useful content from each input frame conditioned on the neighborhood information. In addition, We integrate the neighborhood attention mechanism into an RNN-based encoder-decoder framework to capture the spatial-temporal structure in a video which is consistent with that in the neighborhood. Experiments on three widely-used benchmark datasets demonstrate superior performance of our proposed approach over state-of-the-arts and also validate the effectiveness of our proposed neighborhood attention mechanism. There are several future works to do. Firstly, we can integrate the neighborhood attention mechanism into more delicate architectures such as hierarchical LSTM to further improve the performance. Besides, we will consider optimizing the binary codes and spatial-temporal features which are used to calculate the similarity structure in the video space in an alternating way.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61822603, Grant U1813218, Grant U1713214, Grant 61672306, Grant 61572271, Grant 41876098 and Grant 61806110, in part by the Shenzhen Science and Technology Project under Grant JCYJ20151117173236192, in part by the National Postdoctoral Program for Innovative Talents under Grant BX201700137, in part by the China Postdoctoral Science Foundation under Grant 2018M630159.

References

[1] 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.

[2] 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.

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

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

[5] 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.

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

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

[8] 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.

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

[10] 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.

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] Zhixiang Chen, Jiwen Lu, Jianjiang Feng, and Jie Zhou. Nonlinear structural hashing for scalable video search. *TCSVT*, 28(6):1421–1433, 2017.

[21] Chao Ma, Yun Gu, Wei Liu, Jie Yang, and Xiangjian He. Unsupervised video hashing by exploiting spatio-temporal feature. In *ICONIP*, pages 511–518, 2016.

[22] Yun Gu, Chao Ma, and Jie Yang. Supervised recurrent hashing for large scale video retrieval. In *ACM'MM*, pages 272–276, 2016.

[23] 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.

[24] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *TPAMI*, 40(4):769–790, 2018.

[25] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - A survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

[26] Kyung-Min Kim, Seong-Ho Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. Multimodal dual attention memory for video story question answering. In *ECCV*, pages 698–713, 2018.

[27] 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.

[28] 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.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[30] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[31] 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.

[32] Adam Santoro, Ryan Faulkner, David Raposo, Jack W. Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy P. Lillicrap. Relational recurrent neural networks. In *NIPS*, pages 7310–7321, 2018.

[33] 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.

[34] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[35] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.

[36] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, pages 557–566, 2015.

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

[38] Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[46] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

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

[48] Matthieu Courbariaux and Yoshua Bengio. Binarynet: training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[49] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, pages 1029–1038, 2016.

[50] Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, and Hervé Jégou. How should we evaluate supervised hashing? In *ICASSP*, pages 1732–1736, 2017.

[51] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015.