Unsupervised Variational Video Hashing with 1D-CNN-LSTM Networks

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

***Abstract*—Most existing unsupervised video hashing methods generate binary codes by using RNNs in a deterministic manner, which fails to capture the dominant latent variation of videos. In addition, RNN-based video hashing methods suffer the content forgetting of early input frames due to the sequential processing inherency of RNNs, which is detrimental to global information capturing. In this work, we propose an unsupervised variational video hashing (UVVH) method for scalable video retrieval. Our UVVH method aims to capture the salient and global information in a video. Specifically, we introduce a variational autoencoder to learn a probabilistic latent representation of the salient factors of video variations. To better exploit the global information of videos, we design a 1D-CNN-LSTM model. The 1D-CNN-LSTM model processes long frame sequences in a parallel and hierarchical way, and exploits the correlations between frames to reconstruct the frame-level features. As a consequence, the learned hash functions can produce reliable binary codes for video retrieval. We conduct extensive experiments on three widely used benchmark datasets, FCVID, ActivityNet and YFCC to validate the effectiveness of our proposed approach.**

***Index Terms*—hashing, scalable video retrieval, unsupervised, variational.**

I. Introduction

Over the past few years, overwhelming visual data has been exploding over Internet. For example, more than 300 hours of videos are being uploaded on the YouTube video website per minute. Dramatically increasing visual data brings enormous challenges to visual retrieval and also makes large-scale visual retrieval an urgent need. Even though there are extensive studies on scalable image retrieval [1]–[12], only a few works focus on scalable video retrieval [13]–[18]. Compared with images, videos provide rich visual patterns as well as correlations between frames, which makes video retrieval more sophisticated than image retrieval [19], [20].

Confronting large scale datasets and high dimensional features, video hashing methods have been widely applied in scalable video retrieval [13]–[18]. Among them, unsupervised video hashing methods which do not require time-consuming manual labels are more practicable for most realistic problems. In general, unsupervised video hashing methods integrate data properties such as data distribution and manifold structure to map resemble data into resemble binary codes for efficient video retrieval [15], [17], [21]. Usually the salient factors are important to video retrieval among rich patterns contained in a video such as various illumination and textured background [22]. However, most existing unsupervised video hashing methods do not adequately exploit the salient factors of video variations.

Furthermore, most deep video hashing methods [16], [17] employ a CNN-RNN structure to capture the spatial-temporal information in a video. Specifically, they extract the frame-level features by using a pre-trained CNN network and encode them to a video level representation by using a long-short term memory (LSTM) network [23], [24]. Although LSTM is widely used for processing sequence data, its sequential mechanism leads to the fact that the early input frames are more likely to be forgotten. Hierarchical LSTM attempts to alleviate the content forgetting disadvantage by reducing the input length of a long video [25]. Whereas it still sets priority to the latest input frames, which deteriorates the global information capturing and leads to suboptimal binary codes.

In this work, we propose a 1D-CNN-LSTM based Unsupervised Variational Video Hashing (UVVH) method for scalable video retrieval. UVVH has two strengths over most existing video hashing methods. 1) Instead of learning in a fully deterministic way, UVVH integrates the variational mechanism to capture the salient factors of video variations. This is beneficial to out-of-sample extension. 2) The 1D-CNN-LSTM model learns the binary code in a parallel and hierarchical manner, and exploits the correlations between frames to reconstruct frame features. This model can better exploit compositional structure in the video, thus learns reliable hash functions. As shown in Fig. 1, our proposed UVVH is in a variational encoder-decoder framework. We design a variational 1D-CNN encoder to learn a posterior distribution for each video conditioned on input frame-level features. Through this variational mechanism, the binary code is encoded in a probabilistic manner from the posterior distribution. To better exploit the global information, we propose a 1D-CNN-LSTM decoder to reconstruct the frame-level features from the binary code. Specifically, a 1D-deconvolution decoder decodes the binary code in parallel and yields intermediate frame features. And a LSTM decoder generates the final frame features from the binary code and the intermediate features sequentially by further exploiting the correlations between frames. We train

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, 61806110, 61822603, U1713214, 61672306, 61572271, and 61527808, the National Postdoctoral Program for Innovative Talents under Grant BX201700137, and China Postdoctoral Science Foundation under Grant 2018M630159.

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.

Zhixiang Chen, 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: zxchen@tsinghua.edu.cn; lujiwen@tsinghua.edu.cn; jzhou@tsinghua.edu.cn.

Corresponding author: Xiu Li.

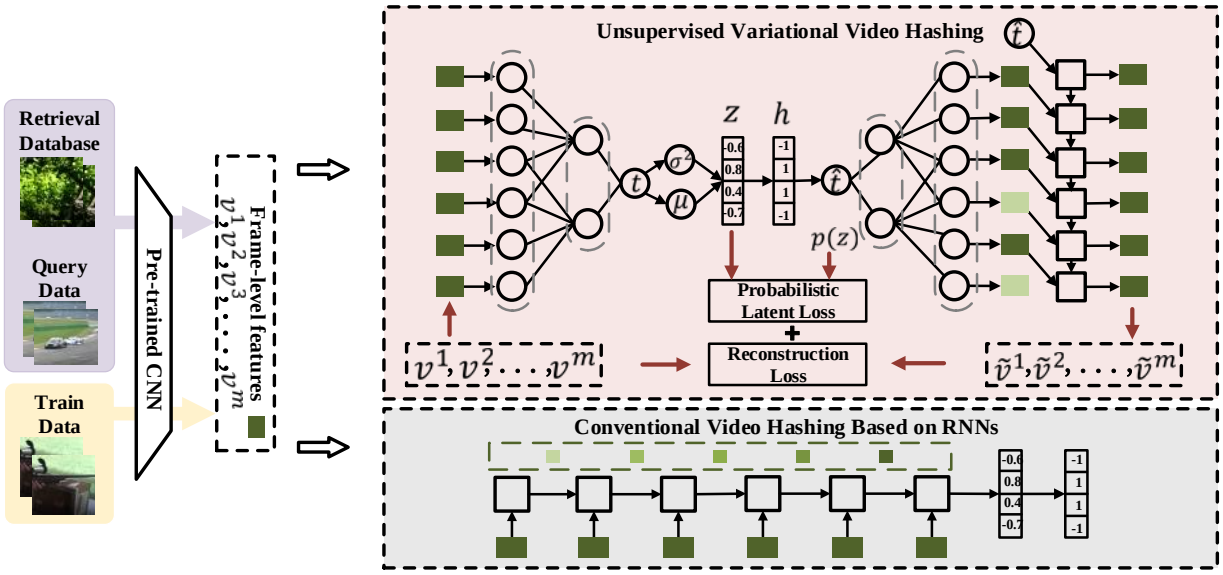


Fig. 1: The overview of the proposed Unsupervised Variational Video Hashing (UVVH) for scalable video retrieval. Unlike conventional video hashing methods (on the bottom) which employ a RNN encoder to learn the binary code in a deterministic manner, UVVH (on the top) integrates the variational 1D-CNN encoder and the 1D-CNN-LSTM decoder to learn hash functions. The variational 1D-CNN encoder takes frame-level features as inputs and generates a binary code. Then the 1D-CNN-LSTM decoder reconstructs the frame-level features from the binary code. The whole network is trained with a probabilistic latent loss and a reconstruction loss. The hollow squares denote LSTM units. The gray dashed frames denote convolutional/deconvolutional layers. The green rectangles denote frame-level features. Lighter green means that less information of the input frame is preserved. The green dashed frame denotes the content forgetting of early frame inputs.

the whole model with a probabilistic latent loss and a reconstruction loss. The former encourages the learned posterior distribution to be close to a pre-defined prior and the latter encourages the input frame-level features to be reconstructed. Extensive experimental results on three public video datasets demonstrate the effectiveness of UVVH. The contributions of this work are briefly concluded as follows:

1) We integrate the variational mechanism to capture the salient factors of video variations. By learning hash codes in a generative form, the hashing model is capable to estimate the statistics of training data and is suitable to out-of-sample extension.

2) We design 1D-CNN-LSTM model, which integrates the advantage of 1D-CNN and LSTM to better capture the global information in videos. By processing the input frames in a hierarchical and parallel manner, it alleviates information forgetting that LSTM-based models usually suffer. And compared with fully convolutional models, 1D-CNN-LSTM model can better exploit the correlations between frames.

3) We conduct extensive experiments to demonstrate the superior performance of UVVH over state-of-the-art methods and also validate the effectiveness of the variational mechanism and 1D-CNN-LSTM model.

II. Related Work

In this section we briefly review three related topics: 1) learning-based hashing, 2) video representation learning and 3) variational autoencoder.

A. Learning-based Hashing

Learning-based hashing aims to learn a series of hash functions from data so that good retrieval performance could be achieved in the Hamming space [26], [27]. Learning-based hash methods can be generally classified into shallow hashing and deep hashing. Shallow hashing methods take hand-crafted features as inputs and learn hash functions from them [1], [2], [5]. The most representative one is IteraTive Quantization (ITQ) proposed by Gong *et al.* [1]. It found a rotation of zero-centered data to minimize the quantization error when mapping the data to vertices of the binary hypercube.

In recent years, a variety of deep learning algorithms have been applied to hash learning [28]–[35]. Among them, Liong *et al.* [28] and Lai *et al.* [30] incorporated pair-wise supervision and triplet ranking loss respectively to train the deep hashing model. Chen *et al.* [34] transformed the original binary optimization into differentiable optimization problem over hash functions through series expansion to deal with the objective discrepancy caused by relaxation.

There are also some works focusing on video hashing [13], [16]–[18], [21], [36]–[38]. For example, Song *et al.* [16] proposed Multiple Feature Hashing (MFH) to explore the local structural information, however they were unaware of the temporal order of video frames. Wu *et al.* [39] proposed Unsupervised Deep Video Hashing (UDVH) which aimed to balance the variation of each dimension of the hash code. Qi *et al.* [38] proposed 3DCNN-based hash which attempted

to capture the motion information through multiple adjacent frames. The limitation is that 3DCNN can only cope with short video clips of around 16 frames [40]. Zhang *et al.* [17] proposed Self-Supervised Temporal Hashing (SSTH) which attempted to capture the temporal information in a video. The long processing path from inputs to outputs makes it hard to capture the long-range correlations in a video. Song *et al.* [37] extended SSTH into Self Supervised Video Hashing (SSVH) via exploiting more powerful hierarchical LSTM networks. While the hierarchical structure could reduce the input length, it still deals with the input frames in a sequential manner. Li. *et al.* [21] learnt temporal representation and appearance representation simultaneously, which also exploited a LSTM-based model.

B. Video Representation Learning

RNNs have been widely employed to learn video representation in recent years [25], [41]–[46]. For example, Donahue. *et al.* [41] introduced Long-term Recurrent Convolutional Networks (LRCN) which utilized CNN-LSTM structure to extract saptial-temporal representation for each frame. In their work, CNN was used for frame-level features extraction instead of video-level representation generation. Srivastava *et al.* [43] proposed a RNN based autoencoder to learn the video representation. Venugopalan *et al.* [44] proposed a two-layer stacked LSTM to process the frame sequence. Pan *et al.* [25] introduced hierarchical LSTM layer which could exploit longer temporal structure by reducing the length of input information flow. RNNs set priority to latest input frames and are likely to forget the early ones, thus conventional RNN-based methods are not suitable for global information capturing.

CNN, which has shown promising ability to capture the semantic information as well as the correlations between sequence fragments [47], [48], is also an important branch to learn video representation. For example, Karpathy *et al.* [47] proposed an architecture with two spatial resolutions channels to boost spatial-temporal pooling. The works contemporaneous with ours [49]–[51] employed 1D-CNN to capture the video representation. Among them, Kim *et al.* [49] applied 1D-CNN to aggregate the learned video concepts in a video story QA model. Guo *et al.* [51] proposed a fully convolutional network to identify multi-scale temporal action proposals. They utilized only the temporal convolutions to retrieve accurate action proposals for video sequences. Rochan *et al.* [50] proposed a fully convolutional autoencoder for video summarization. However, CNN-based encoder-decoder models are difficult to reconstruct the frames in a fully feedforward manner.

C. Variational Autoencoder

Variational autoencoder (VAE) [52], [53], which reformulates the auto-encoder model as a variational inference problem, has been widely used to generate a variety of complex data and representations [54]–[59]. Kingma *et al.* [52] and Rezende *et al.* [53], who first proposed VAEs, showed that VAEs could well exploit salient factors of variation by capturing the latent representation in an unsupervised and probabilistic manner. Yang *et al.* [56] proposed an improved VAE with dilated convolutions for generative text modeling. Walker *et al.* [58] proposed a conditional VAE to predict the dense trajectory of pixels in a scene. Pu *et al.* [59] developed a novel variational autoencoder to model images, as well as associated labels or captions.

Promising performance as VAE has shown in a variety of fields, only a few works attempted to learn hash functions by using variational mechanism. Liong *et al.* [60] proposed cross-model deep variational hashing method for cross-modality retrieval. They learned hashing mapping in a deterministic manner, and then modeled a probabilistic latent variable to approximate the inferred binary codes. They claimed that in this way, the model could be more general and suitable for out-of-sample extension. Chaidaroon *et al.* [61] proposed variational deep semantic hashing for text documents, where the binary code generation and reconstruction were conducted through a series of fully connected (FC) layers. Dai *et al.* [62] proposed a generative approach to learn hash functions through Minimum Description Length principle while referring to VAE to address an expensive integer programming subproblem. Shen *et al.* [63] proposed Deep Variational networks for Binary representation learning (DBV). Since DBV is initially designed for image hashing, directly extending it to video hashing will inevitably lead to performance degradation.

III. Unsupervised Variational Video Hashing

In this section, we first present the overall framework of UVVH in subsection III-A. The framework of UVVH is in a variational encoder-decoder mode. We detail the variational 1D-CNN encoding network in subsection III-B. And we describe the 1D-CNN-LSTM decoding network in subsection III-C.

A. Overall Framework of UVVH

Let $\boldsymbol{S} = \{\boldsymbol{S}_i\}_{1=1}^{N}$ be a collection of N videos, among which $\boldsymbol{S}_i$ denotes the i-th video. For each video, we uniformly sample m frames. We process each frame via a conventional CNN to gain a frame-level feature. In this way, we transform the raw video $\boldsymbol{S}_i$ to a sequence of frame-level features $\{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, .., \boldsymbol{v}_i^m\}^T \in \mathbb{R}^{m\times l}$. $\boldsymbol{v}_i^j$ denotes the j-th frame-level feature of the i-th video with dimension of l. We denote the frame-level features as $\{\boldsymbol{v}_i^j\}_{j=1}^m$ for short.

Our goal is to learn a mapping $\mathcal{B}$ to transfer each frame-level feature sequence $\{\boldsymbol{v}_i^j\}_{j=1}^m$ to a compact binary code $\boldsymbol{h}_i \in \{-1, 1\}^k$, such that the similarity structure between videos is well preserved in the Hamming space. k is the code length. We present the nonlinear mapping $\mathcal{B}$ as follows:

$$\mathcal{B} : \mathbb{R}^{m\times l} \rightarrow \{\pm 1\}^k. \tag{1}$$

UVVH composes of a variational 1D-CNN encoder which generates the binary code by learning an approximate posterior distribution and a 1D-CNN-LSTM decoder which reconstructs the frame-level features from the binary code. In order to preserve the similarity structure in Hamming space for scalable video retrieval, the encoder-decoder model is trained to capture

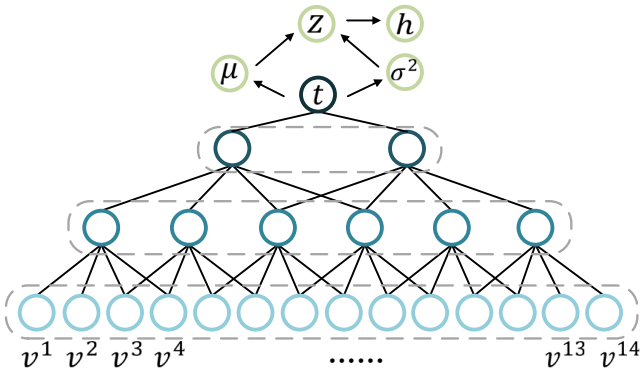


Fig. 2: The 1D-CNN variational encoder. Frame-level features are the inputs, and the binary code is the output. Taking input frame length of 14 for example, there are three 1D-convolutional layers. The kernel sizes for these layers are 4, 4 and 2 respectively. The strides for all these layers are 2. A single real-value vector $\boldsymbol{t}_i$ is generated via the 1D-convolution network. Then the mean and variance of a Gaussian distribution are calculated from $\boldsymbol{t}_i$. Next, a real-value vector $\boldsymbol{z}_i$ is sampled from the Gaussian distribution. Finally, the binary code $\boldsymbol{h}_i$ is obtained by discretizing $\boldsymbol{z}_i$. Gray dashed frames denote 1D-convolutional layers. Without confusion, the subscripts for all the notations i are omitted in the figure.

the salient factors of a video and reconstruct the frame features. The loss function of UVVH composes of two terms as follows:

$$L = \alpha_1 L_{prob} + \alpha_2 L_{recon}. \tag{2}$$

L_{prob} is a probabilistic latent loss between the approximate posterior distribution learned by the variational encoding network and a predefined prior. Minimizing L_{prob} encourages the approximate posterior to be close to the prior. The calculation of L_{prob} is described in subsection III-B. L_{recon} is a reconstruction loss between the reconstructed frame features and input frame-level features. Minimizing L_{recon} encourages the decoding network to reconstruct the frame-level features from the binary code. The calculation of L_{recon} is described in subsection III-C. α_1 and α_2 are hyper-parameters which balance these two losses.

B. Variational Encoding Network

We design a variational encoding network to learn an approximate posterior distribution in order to capture the salient information of the input video. The objective is to minimize the divergence between the posterior distribution and a predefined prior.

The variational 1D-CNN encoding network is depicted in Fig. 2. The inputs of the encoding network are frame-level features $\{\boldsymbol{v}_i^j\}_{j=1}^m$ of a video $\boldsymbol{S}_i$. At each layer, 1D-convolution is operated along the length dimension of the feature sequence with d_a 1D-convolution kernels $\{\boldsymbol{W}|W_j \in \mathbb{R}^{d_b \times k_s},\ j = 1, 2, ..d_a\}$. d_b is the feature dimension of the layer's input, k_s is the kernel size, d_a is the number of kernel channels and j denotes the j-th channel. With d_a kernels operated at each layer, the output features of the corresponding layer are projected to d_a-dimension: $\mathbb{R}^{d_b} \rightarrow \mathbb{R}^{d_a}$. The output features at each layer are then injected to the next layer. With 1D-convolution operation, the length of the outputs at each 1D-convolutional layer decreases gradually and a d-dimensional real-value vector $\boldsymbol{t}_i$ is finally obtained. We denote the mapping from frame-level features $\{\boldsymbol{v}_i^j\}_{j=1}^m$ to $\boldsymbol{t}_i$ as $\mathcal{F}$ and present the mapping as follows:

$$\boldsymbol{t}_i = \mathcal{F}(\{\boldsymbol{v}_i^j\}_{j=1}^m, \theta), \tag{3}$$

where θ denotes the learnable parameter set of the encoding network.

Since variational mechanism can improve the robustness and generalisation for representation learning, we integrate it into the encoding network to capture holistic and salient information in a video. We aim to obtain a posterior distribution $p(\boldsymbol{z}_i|\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, .., \boldsymbol{v}_i^m)$, latent variable $\boldsymbol{z}_i$ sampled from which can recover the frame-level features $\{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, .., \boldsymbol{v}_i^m\}^T$. $\boldsymbol{z}_i$ is a real-value latent variable with dimension of k. Since the frame-level features have been encoded to $\boldsymbol{t}_i$, we rewrite this posterior distribution as $p(\boldsymbol{z}_i|\boldsymbol{t}_i)$ for short. Since introducing a nonlinear mapping from the latent variable $\boldsymbol{z}_i$ to the frame features results in intractable posterior distribution $p(\boldsymbol{z}_i|\boldsymbol{t}_i)$ [53], we learn an approximation $q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)$ for the true posterior distribution. To enable a high capacity, we assume the posterior $q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)$ to be a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2))$, where $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ are the mean and variance of $\boldsymbol{z}_i$ respectively. The calculations of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are presented as follows:

$$\boldsymbol{\mu}_i = dense(tanh(\boldsymbol{t}_i), k), \tag{4}$$

$$\log \boldsymbol{\sigma}_i = dense(tanh(\boldsymbol{t}_i), k), \tag{5}$$

where $tanh$ is an activate function which is defined as: $tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. $dense(\boldsymbol{x}, k)$ is a linear function that maps the vector $\boldsymbol{x}$ to a k-dimensional vector. It is presented in detail as $dense(\boldsymbol{x}, k) = \boldsymbol{x} \times \boldsymbol{W} + \boldsymbol{b}$. $\boldsymbol{W} \in \mathbb{R}^{d \times k}$ is a learnable parameter matrix and $\boldsymbol{b} \in \mathbb{R}^k$ is a learnable bias vector. It is worth to mention that these two $dense$ functions in (4) and (5) do not share parameters.

Similar to [53], we let the prior over the latent variables be the centered isotropic multivariate Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We encourage the approximate posterior distribution to be close to the prior with Kullback-Leibler (KL) Divergence D_{KL}. We derive the analytic form of the KL Divergence as follows:

$$\begin{aligned} &D_{KL}(q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)||p(\boldsymbol{z}_i)) \\ &= \int q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)(\log q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i) - \log p(\boldsymbol{z}_i))d\boldsymbol{z}_i \\ &= -\frac{1}{2}\sum_{j=1}^{k}(1 + log\boldsymbol{\sigma}_{ij}^2 - \boldsymbol{\mu}_{ij}^2 - \boldsymbol{\sigma}_{ij}^2), \end{aligned} \tag{6}$$

where j denotes the j-th hash function. We have the specific formulation of L_{prob} in (2) as:

$$L_{prob} = -\frac{1}{N}\sum_{i=1}^{N} D_{KL}(q(\boldsymbol{z}_i|\boldsymbol{t}_i)||p(\boldsymbol{z}_i)). \tag{7}$$

While the loss in (7) is related to $p(\boldsymbol{z}_i)$ and $q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)$,

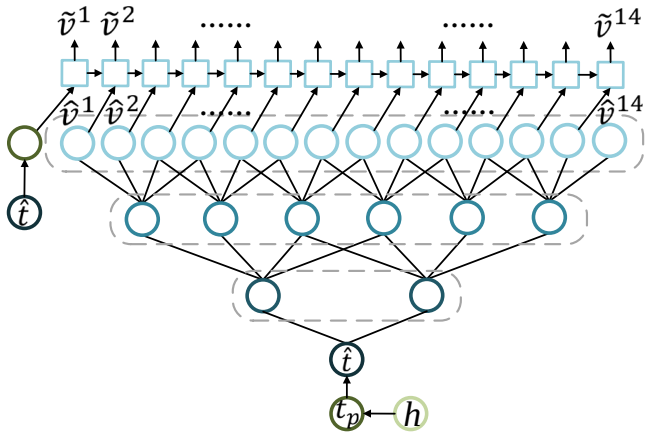


Fig. 3: The 1D-CNN-LSTM decoder. The input of the decoder is the binary code $\boldsymbol{h}_i$ and the outputs are final reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^{14}$. The binary code $\boldsymbol{h}_i$ is projected to a real-value vector $\hat{\boldsymbol{t}}_i$ with two FC layers. The 1D-deconvolution network generates intermediate frame features $\{\hat{\boldsymbol{v}}_i^j\}_{j=1}^{14}$ from the real-value vector $\hat{\boldsymbol{t}}_i$. These intermediate reconstructed vectors, together with $\hat{\boldsymbol{t}}_i$, are input to the LSTM decoding network to generate final reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^{14}$. Gray dashed frames denote 1D-deconvolutional layers. Blue squares denote LSTM units. Without confusion, the subscripts for all the notations i are omitted in the figure.

the generation of the binary code requires extra sampling process. In the training stage, we sample $\boldsymbol{z}_i$ from the posterior distribution $q_\theta(\boldsymbol{z}_i|\boldsymbol{t}_i)$ as follows:

$$\begin{aligned} \boldsymbol{z}_i =& \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \epsilon_i, \\ & \epsilon_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \end{aligned} \tag{8}$$

Then we discretize $\boldsymbol{z}_i$ to a binary code $\boldsymbol{h}_i$:

$$\boldsymbol{h}_i = sign(\boldsymbol{z}_i), \tag{9}$$

where $sign(x) = 1$ if $x \geq 0$ and $sign(x) = -1$ otherwise. To train the encoder with back-propagation, the derivative should be computed. The derivative of $sign(x)$ is denoted as $sign'(x) = 2\delta(x)$. Since the derivative is zero almost everywhere, we refer to BinaryNet [31] to handle the ill-pose gradient problem. During back-propagation, we set the derivative of $sign(x)$ to be 1 if x is within $[-1, 1]$, and 0 otherwise.

In the testing stage, given a video of retrieval database or a query video $\boldsymbol{S}_q$, the encoder generates the posterior distribution $q_\theta(\boldsymbol{z}_q|\boldsymbol{S}_q) = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2)$ according to Equation (4). Then we obtain $\boldsymbol{z}_q$ in a deterministic manner with $\boldsymbol{z}_q = \boldsymbol{\mu}_q$. This means that in the retrieval stage, we sample $\boldsymbol{z}_q$ at the mean of the approximate posterior distribution, where the sampled presentation is most likely to contain holistic information. Then the binary code $\boldsymbol{h}_q$ is generated by $\boldsymbol{h}_q = sign(\boldsymbol{z}_q)$.

C. 1D-CNN-LSTM Decoding Network

The decoding network is to reconstruct the frame features from the binary code, which is presented as follows:

$$\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m = \mathcal{D}(\boldsymbol{h}_i, \phi), \tag{10}$$

where $\mathcal{D}$ denotes the decoding mapping, ϕ denotes the parameter set of the decoding network, $\boldsymbol{h}_i$ is the binary code, and $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m$ are reconstructed frame features.

As it is shown in Fig. 3, we cascade the 1D-deconvolution network (also known as transposed convolutions [64] and fractionally strided convolutions [65]) and the LSTM network together as the decoding network, aiming to reconstruct the frame-level features from the binary code. The reconstruction process is briefly concluded as follows. We project the binary code $\boldsymbol{h}_i$ to a real-value vector $\hat{\boldsymbol{t}}_i$ as the input of the deconvolution network. The dimension of $\hat{\boldsymbol{t}}_i$ is d, the same as that of $\boldsymbol{t}_i$ in Equation (3). Then the 1D-deconvolution network generates a sequence of intermediate reconstructed frame features $\{\hat{\boldsymbol{v}}_i^j\}_{j=1}^m$, where $\hat{\boldsymbol{v}}_i^j$ is the j-th intermediate reconstructed frame feature of the i-th video. Finally the LSTM network generates the final frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m$.

Instead of direct projection from $\boldsymbol{h}_i$ to $\hat{\boldsymbol{t}}_i$, we first map the binary code to a high-dimensional intermediate real-value vector $\boldsymbol{t}_{pi}$. The dimension of $\boldsymbol{t}_{pi}$ is p which is much higher than d. Then we project $\boldsymbol{t}_{pi}$ to $\hat{\boldsymbol{t}}_i$. This allows the decoding network to recover from the discretization bottleneck [66]. We present the projections as follows:

$$\boldsymbol{t}_{pi} = tanh(dense(\boldsymbol{h}_i, p)), \tag{11}$$

$$\hat{\boldsymbol{t}}_i = tanh(dense(\boldsymbol{t}_{pi}, d)). \tag{12}$$

The deconvolution operation can be viewed as a reverse process of the convolution operation. With deconvolution operation, the length of the sequence progressively increases through each deconvolutional layer. Then the 1D-deconvolution network outputs intermediate frame features $\{\hat{\boldsymbol{v}}_i^j\}_{j=1}^m$. While the 1D-CNN network does not suffer content forgetting of early input frames by parallel processing, it is hard to reconstruct the frame-level features due to the feed-forward structure. Specifically, the 1D-deconvolution network reconstructs the frame-level features by deriving the following conditional distribution:

$$P(\hat{\boldsymbol{v}}_i|\boldsymbol{h}_i) = P(\hat{\boldsymbol{v}}_i^1, \hat{\boldsymbol{v}}_i^2, ...\hat{\boldsymbol{v}}_i^m|\boldsymbol{h}_i). \tag{13}$$

This distribution contains the assumption that the reconstructed frame features are only conditioned on $\boldsymbol{h}_i$. With this assumption, the decoding network has to capture most details of each frame to accurately recover the frame-level features, which is quite difficult.

Under most circumstances, there are strong correlations between frames within a video. Specifically, as long as a sequence of frames have been observed, the following frames can be partly implicated from them. In order to exploit the correlations between frames, we integrate an LSTM decoder to further reconstruct the frame features from the intermediate reconstructed frame features and obtain the final reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m$. The decoding process of the LSTM network can be modeled as the following full joint distribution:

$$P(\tilde{\boldsymbol{v}}_i^1, \tilde{\boldsymbol{v}}_i^2, ...\tilde{\boldsymbol{v}}_i^m|\boldsymbol{h}_i) = \prod_{j=1}^m P(\tilde{\boldsymbol{v}}_i^j|\boldsymbol{h}_i, \hat{\boldsymbol{v}}_i^1, ..\hat{\boldsymbol{v}}_i^{j-1}). \tag{14}$$

By doing so, when some of the frame features are not successfully recovered by the 1D-deconvolution decoder, the

Algorithm 1 Training of UVVH

Input: Training set $\{\boldsymbol{S}_i\}_{i=1}^N$, network learning parameters, iterative number $Iter$, objective function parameters α_1, α_2.
Output: Network parameters θ and ϕ.
Step 1 Initialization:
Uniformly sample m frames for each video.
Extract frame-level features $\{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, .., \boldsymbol{v}_i^m\}_{i=1}^N$ by using pre-trained CNN.
Initialize UVVH network parameters.
Step 2 UVVH network learning:
for $iter = 1, 2, .., Iter$ **do**
  for $i = 1, 2, .., N$ **do**
    %Forward Propagation:
    Compute $\boldsymbol{t}_i$ for each input video $\boldsymbol{S}_i$ according to (3).
    Calculate $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ with (4) and (5).
    Sample $\boldsymbol{z}_i$ with (8).
    Obtain binary code $\boldsymbol{h}_i$ with (9).
    Compute the real-value vector $\hat{\boldsymbol{t}}$ from $\boldsymbol{h}$ with (11).
    Generate the intermediate frame features $\{\hat{\boldsymbol{v}}_i^j\}_{j=1}^m$ by the decovolutiona decoder.
    Generate the final frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m$ by the LSTM decoder.
    %Backward Propagation:
    Compute gradient of loss function with (2), (7), (15).
    Perform gradient descent to learn θ and ϕ.
  end for
end for
Return: learned network parameters θ and ϕ.

LSTM decoder can still well recover the frame features by exploiting the correlations between frames. Therefore, we can relief the decoding network from focusing too much details and allow it to pay more attention on global and holistic information.

Inspired by caption models [25] [67], we design the LSTM decoding process as follows. At first time step, we project $\hat{\boldsymbol{t}}_i$ to frame-level feature space and inject it into LSTM. Here, $\hat{\boldsymbol{t}}_i$ is assumed to contain full information from the binary code $\boldsymbol{h}_i$ and it provides global guiding to the LSTM decoder. Then the LSTM decoder yields the first reconstructed frame feature $\tilde{\boldsymbol{v}}_i^1$. Next, we inject the first intermediate reconstructed frame feature $\hat{\boldsymbol{v}}_i^1$ into LSTM to obtain the second frame feature $\tilde{\boldsymbol{v}}_i^2$. We conduct similar operations recurrently till we obtain the m-th reconstructed frame-level feature $\tilde{\boldsymbol{v}}_i^m$.

We use Mean-Square Error (MSE) to describe how well each video is reconstructed. Then we have the specific formulation of the reconstruction loss L_{recon} in (2):

$$L_{recon} = \frac{1}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} ||\boldsymbol{v}_i^j - \tilde{\boldsymbol{v}}_i^j||_2^2. \tag{15}$$

Minimizing L_{recon} encourages the final reconstructed frame features $\{\tilde{\boldsymbol{v}}_i^j\}_{j=1}^m$ to be close to the input frame-level features $\{\boldsymbol{v}_i^j\}_{j=1}^m$. The training process of UVVH is summarized in Algorithm 1.

IV. Experiments

To validate the effectiveness of our proposed UVVH method for scalable video retrieval, we conduct extensive experiments on three large-scale video datasets: FCVID [68], ActivityNet [69] and YFCC [70]. We conduct all the experiments with Pytorch on single Geforce GTX 1080 Ti GPU.

A. Datasets

1) FCVID. Fudan-Columbia Video Dataset 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. The categories in FCVID cover a wide range of topics like procedural events (e.g.,"brushing teeth"), social events (e.g., "tailgate party"), objects (e.g., "bee"), scenes (e.g., "tornado"), etc. Owing to the damaged data as well as category overlap, there are 91,185 videos available. Following the setting in [17], we utilize 45,585 videos as training set and 45,600 videos as retrieval database and queries.

2) ActivityNet. This recently released video dataset covers a wide range of complex human activities. It 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. Many of the videos in this dataset are shot by amateurs in uncontrolled environments, where the variances within the same activity category are often large. Since the test split of ActivityNet is not publicly available, we use validation set as our test set. In each category, we randomly choose 1000 videos from the validation set for queries, and use the remaining validation videos for retrieval database. In conclusion, we use 9,722 videos for training, 1,000 for queries and 3,760 for retrieval database.

3) YFCC. Yahoo Flickr Creative Commons 100 Million Dataset is the largest public video dataset which contains 0.8M videos. In the labeled split, there are 80 categories collected from the third level of MIT SUN scene hierarchy [71]. We exploit 511,044 videos for our experiment. We use 409,788 unlabeled videos for training. Among the 101,256 labeled videos, we randomly choose 1,000 videos with non-zero label as queries and the rest as retrieval database.

B. Evaluation Metrics

We employ Average Precision at top-K retrieved videos (AP@K) for retrieval performance evaluation [72]. AP@K is defined as follows:

$$\text{AP@K} = \frac{1}{min(R,K) \sum_{i=1}^{K} \frac{R_i}{i} \times I_i}, \quad 1 \le i \le K \tag{16}$$

where R is the number of total relevant videos in the database. R_i is the number of relevant videos in the top-i retrieval result. $I_i = 1$ if the i-th video is considered to belong to the same category with the query and $I_i = 0$ otherwise. We use the mean of AP@K over all the queries (mAP@K) for the main evaluation metric. To provide a detailed observation of the

retrieval performance, we use Precision-Recall curve as an additional evaluation measurement. To sort the results, we rank videos according to their Hamming distance from the query.

C. Implementation Details

On FCVID and YFCC datasets, we follow the setting in [17]. We uniformly sample 25 frames for each video and use the 16 layers VGG network [73] pre-trained on ImageNet [74] to extract the feature for each frame. In this way, we get 4096-dimensional frame-level features as inputs. In other words, the dimension of the input feature sequence is 4096×25. Since [17] did not include ActivityNet dataset for evaluation, we follow the setting in [21] on this dataset. We use the 50 layers ResNet network [75] pre-trained on ImageNet to extract the frame-level features, then we get 2048×25-dimensional input feature sequence.

Our UVVH network is composed of a 1D-convolution encoder, a 1D-deconvolution decoder as well as an LSTM decoder. Both the 1D-CNN encoder and decoder have three convolutional/deconvolutional layers. Each convolutional/deconvolutional layer is followed by a batch normalization layer. We apply the activate function $tanh$ for the output of each convolutional/deconvolutional layer. The length of the feature sequence decreases from 25 to 1 through convolution operations and increases from 1 to 25 through deconvolution operations. We show the architecture details in TABLE I. Without confusing, we omit standard normalization layers and activation operations in this table. Specifically, fc-x denotes a FC layer whose output is x. k is the length of binary code.

TABLE I: Configurations of the UVVH network

type	kernel size/stride/hidden size	output size
1D-conv1	4 / 2 / -	512×10
1D-conv2	4 / 2 / -	512×4
1D-conv3	4 / 2 / -	256×1
fc-$\boldsymbol{\mu}$/ $\log \boldsymbol{\sigma}$	-	k
$sign$	-	k
fc-$\boldsymbol{t}_p$	-	4096
fc-$\hat{\boldsymbol{t}}$	-	256×1
1D-deconv1	4 / 2 / -	512×4
1D-deconv2	4 / 2 / -	512×10
1D-deconv3	4 / 2 / -	$4096/2048 \times 25$
LSTM	- / - / 256	$4096/2048$

We initialize parameters in the network by using Xavier initialization [76] (the weight is set as $\boldsymbol{W} = U[-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}]$ where $\boldsymbol{W} \in \mathbb{R}^{n_{in} \times n_{out}}$). We set the mini-batch size to be 256 and train our model with Adam optimization algorithm [77]. We set the learning rate, the momentum, and weight decay as 0.001, 0.9 and 0.0001 respectively. We set $\alpha_2 = 1$. Considering that the training will suffer the randomness introduced by the monte Carlo sampling procedure on the latent space of VAE, we train our network in two stages. We start training our model in a deterministic manner, which is denoted as stage1. That is, we set $\alpha_1 = 0$ and $\boldsymbol{z} = \boldsymbol{\mu}$ with $\boldsymbol{\mu}$ calculated by (4). After 50 epochs, we continue to train our model in a probabilistic manner, which is denoted as stage2. That is, we set $\alpha_1 = 1$ and sample $\boldsymbol{z}$ over $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with (8). By pre-training the network in stage1, we can get pretty good mean $\boldsymbol{\mu}$ of the posterior Gaussian distribution for stage2 at very beginning. In this way, we can easily sample latent variables which can be quantized into qualified binary codes. We stop training at the 80th epoch. To avoid overfitting, we employ Drop-Out [78] for our decoding network and parameter regularization for the whole network.

D. Compared Methods

We compare our UVVH method with the following methods to validate the effectiveness.

1) **ITQ**. Iterative Quantization [1] is regarded to be a classic unsupervised image hashing method. We extend it for video retrieval. We get a video-level presentation by conducting mean-pooling on the frame-level features.

2) **DH**. Deep Hashing [28] adds a binarization loss function at the top layer of a deep neural network. We extend DH for video retrieval. We apply the RNN encoder-decoder structure in [43] to obtain the video-level representation.

3) **MFH**. Multiple Feature Hashing [16] learns hash functions based on the similarity graph of the frames. It learns frame-level binary codes and then uses average pooling to get the video-level representation. Finally, it binarizes video-level representation to a binary code.

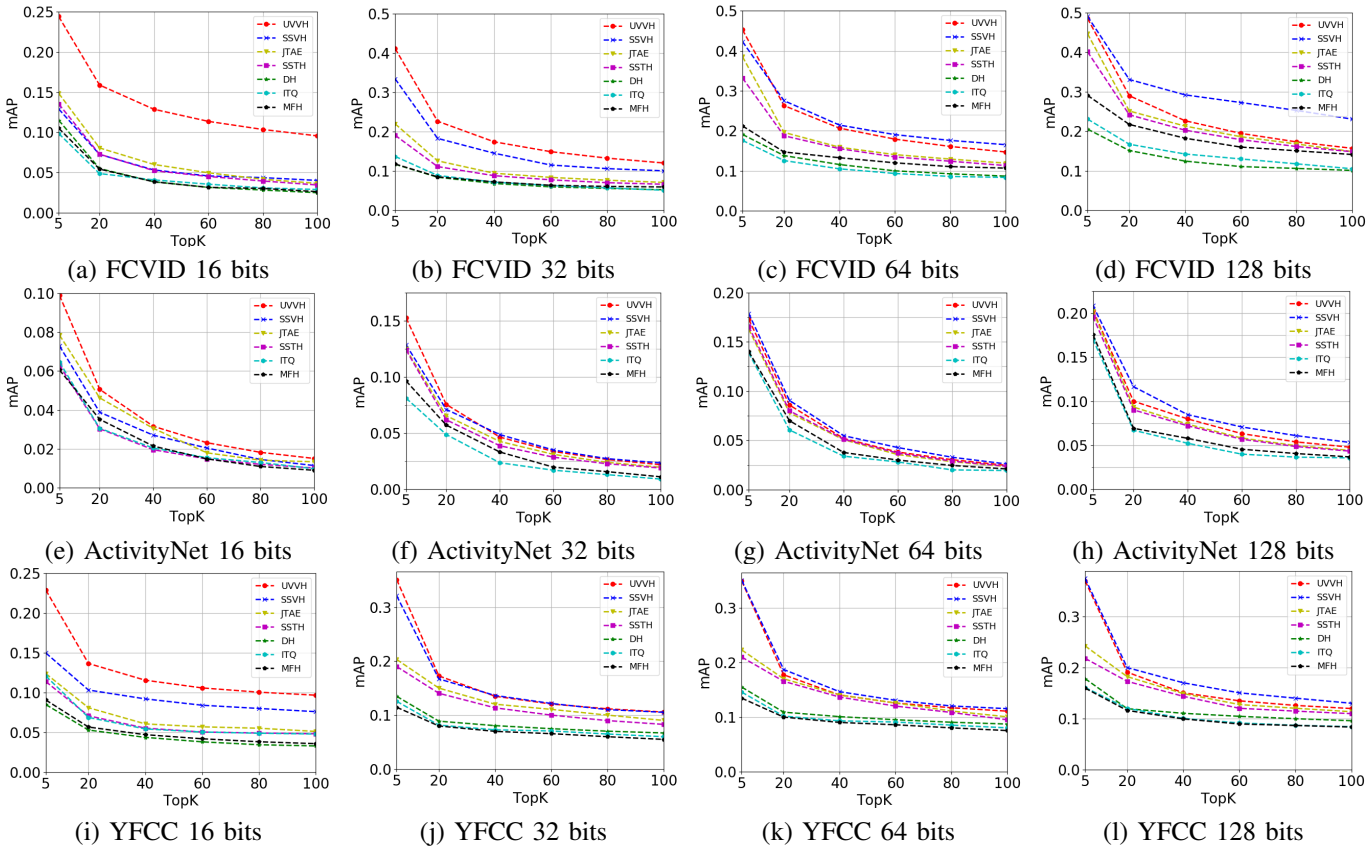
4) **SSTH**. Self-Supervised Temporal Hashing [17] focuses on exploiting temporal information in videos. It introduces Binary LSTM to encode the frame-level features to a binary code. Moreover, it uses a forward LSTM and a backward LSTM to reconstruct the frame-level features from the binary code respectively.

5) **JTAE**. Joint Temporal Appearance Encoder [21] employs LSTM based autoencoder to jointly learn the appearance and temporal information. Beside a forward LSTM to capture the temporal information, it employs FC layers to learn appearance information for each frame.

6) **SSVH**. Self Supervised Video Hashing [37] is an extension of SSTH which exploits powerful hierarchical LSTM networks. Besides, it simultaneously reconstructs the visual content as well as the neighborhood structure of videos.

E. Results and Analysis

Comparisons with state-of-the-arts: On FCVID dataset, UVVH consistently outperforms MFH, ITQ, DH, SSTH and JTAE with all the code lengths in terms of mAP@K as shown in Fig. 4 (a)-(d). UVVH shows overwhelming advantage when code length is relatively short. Specifically, when the code length is 16 bits, UVVH outperforms the most competitive JTAE by 64.8%, 96.5%, 114.0%, 129.1%, 149.0% and 154.3% in terms of mAP@K (K = 5, 20, 40, 60, 80, 100). This demonstrates that UVVH model is more powerful to capture salient and global information, since retrieval with short codes usually requires to capture salient information instead of details considering the amount of information that short codes could contain. Besides, UVVH outperforms SSVH remarkably with code length of 16 bits and 32 bits. In view of mAP@5, UVVH shows advantage over SSVH with code length of 64 bits and is as good as SSVH with 128 bits. It should be noticed that SSVH further reconstructs the neighborhood structure of

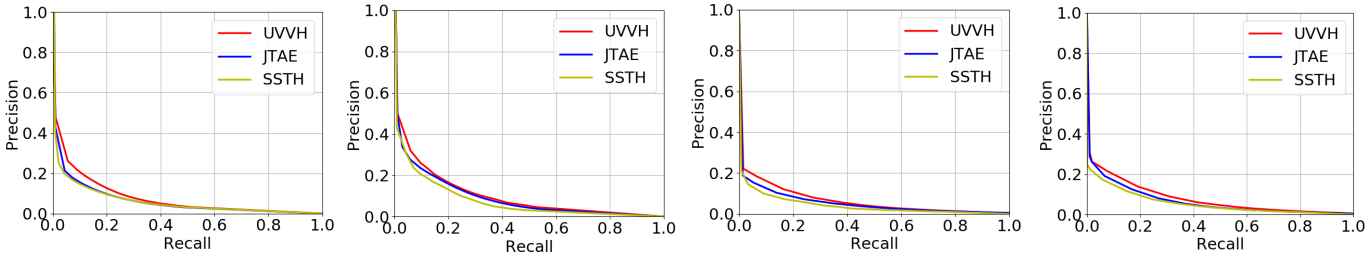


(a) FCVID 16 bits (b) FCVID 32 bits (c) FCVID 64 bits (d) FCVID 128 bits

(e) ActivityNet 16 bits (f) ActivityNet 32 bits (g) ActivityNet 64 bits (h) ActivityNet 128 bits

(i) YFCC 16 bits (j) YFCC 32 bits (k) YFCC 64 bits (l) YFCC 128 bits

Fig. 4: Performance (mAP@K) of different video hashing methods with a variety of code lengths.



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

Fig. 5: Precision-Recall (PR) curves of different video hashing methods with a variety of code lengths. (a) and (b) are the PR curves on FCVID dataset. (c) and (d) are the PR curves on ActivityNet dataset.

videos via an extra neighborhood similarity loss, which is orthogonal with our contributes. This is a possible reason that SSVH outperforms UVVH when K becomes larger with 64 and 128 bits. Another reason is that SSVH employs more complex structure, hierarchical LSTM, therefore can capture more details in the videos. Nevertheless, mAP@5 results of UVVH is very competitive compared with SSVH.

The mAP@K results on ActivityNet are shown in Fig. 4 (e)-(h). In general, the results of all these methods on ActivityNet are poorer than FCVID. This is because the scale of retrieval database is relatively small (3,760 videos in 200 categories), some queries do not have enough true neighbors. Therefore we focus our attention on mAP@5 results. UVVH consistently outperforms MFH, ITQ, SSTH and JTAE with all the code lengths.The advantage of UVVH with short code lengths is also remarkable. When the code length is 16-bit, UVVH outperforms the best competitor JTAE by 44.5% in terms of mAP@5. UVVH outperforms SSVH prominently with code lengths of 16 bits and 32 bits. It has nearly the same performance with SSVH with code lengths of 64 bits and 128 bits.

The mAP@K results on YFCC are shown in Figure 4(i)-(l). As can be seen, UVVH outperforms all the methods except for SSVH with all the code lengths. Specifically, when the code length is 16 bits, UVVH outperforms the most competitive SSVH by 52.6%, 32.2%,25.3%, 25.3% in terms of mAP@K (K = 5, 20, 40, 60). In terms of mAP@5, UVVH outperforms SSVH with code lengths of 16 bits and 32 bits, and has nearly the same performance with SSVH with code lengths of 64 bits and 128 bits.

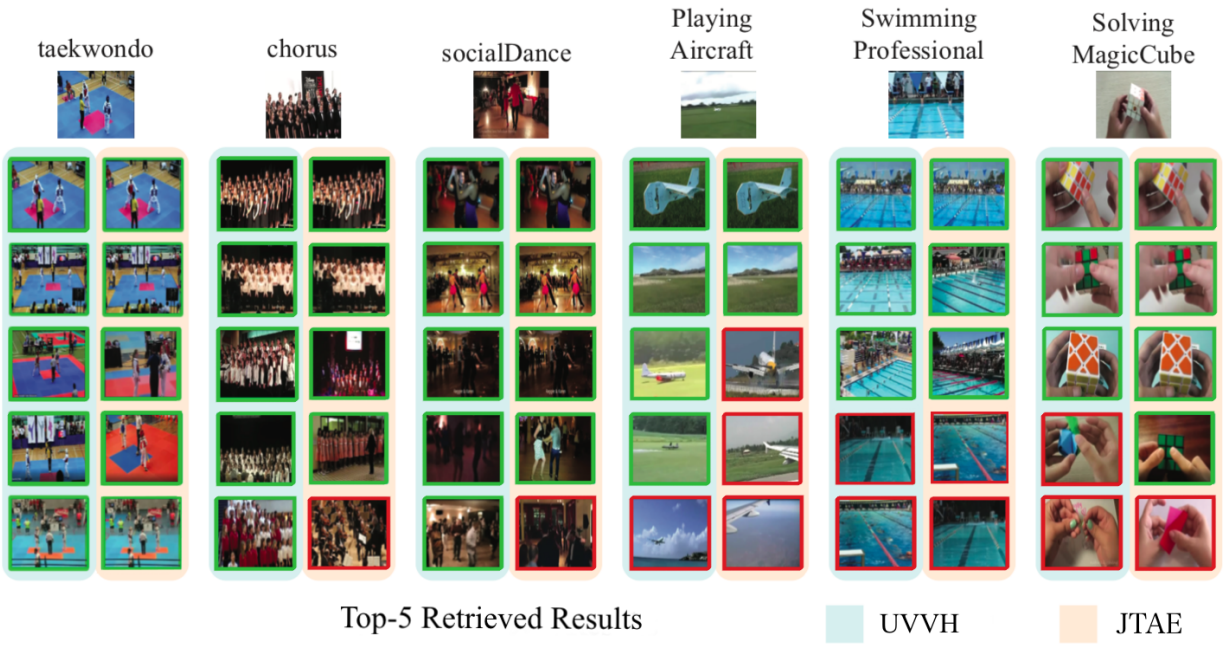


Fig. 6: Top-5 retrieved results on FCVID dataset with code length of 128 bits when using UVVH and JTAE. Frames with green border denote correct retrieved results and frames with red border denote incorrect retrieved results. The first row are six query videos from different categories, and below ones are retrieved videos.

Precision-Recall (PR) curves on FCVID and ActivityNet datasets with code lengths of 32 bits and 64 bits are shown in Fig. 5. The best performance is achieved by UVVH. With the same recall, UVVH consistently achieves higher precision than the other two methods. Consistent with mAP@K results, UVVH shows obvious advantage when the code length is relatively short according to the PR curves.

Cross-dataset evaluation comparisons: It is important to investigate how UVVH generalizes to cross-dataset retrieval tasks, e.g., training on YFCC but testing on FCVID and vice versa. TABLE II lists the cross-dataset (FCVID and YFCC) performance gain (e.g. comparing the mAP@20 when training on YFCC but testing on FCVID with training and testing on FCVID) compared with of several hashing methods. According to TABLE II, all the methods suffer performance drop when training on FCVID and testing on YFCC compared with training and testing on YFCC. And when training on YFCC and testing on FCVID, the performances of all the methods except DH become better compared with training and testing on FCVID. This indicates that smaller training data leads to performance drop and larger training data is conducive to retrieval performance. Furthermore, the performance of UVVH does not fluctuate so much as other methods (MFH, DH and SSTH) during cross-dataset evaluation. This demonstrates that UVVH has better generalization cross different datasets.

Qualitative results: We pick six query videos on FCVID dataset in different categories to show the qualitative results with the top-5 retrieved videos. We present the top-5 retrieved results in Fig. 6 with code length of 128 bits. The left column presents the retrieved results of UVVH and the right column presents that of the comparison method JTAE. As shown in Fig. 6, our proposed UVVH outperforms JTAE in general. For "taekwondo" category, the top-5 retrieved results of both UVVH and JTAE are all correct. For "chorus" and "socialDance" categories, the top-5 retrieved results of UVVH are all correct, but JTAE makes some mistakes. For "socialDance" category, JTAE retrieves a wrong video with content of social meeting. Since the static appearances of social dance videos and social meeting videos are similar (various people in a hall), this may be due to the failure of capturing the temporal information dancing. For "Playing Aircraft", JTAE potentially mistakes passenger plane with playing aircraft. This may be because that JTAE pays more attention to the temporal information flying, but neglects the appearance difference between these two types of planes. In contrast, UVVH shows advantage at capturing global appearance information. For "Solving MagicCude" category, UVVH retrieves two wrong videos which include two hands with colorful stuff. It shows that UVVH mistakes the colorful stuff with a magic cube. A possible reason is that UVVH ignores some useful details of magic cubes. For "Swimming Professional", both UVVH and JTAE are likely to mistake professional swimming with entertainment oriented swimming. One possible reason is that FCVID is a general dataset and does not provide adequate

TABLE II: Cross-dataset mAP@20 gain (%) by Hamming ranking of various methods with code length of 128 bits.

mAP@20-128bits	MFH	ITQ	DH	SSTH	UVVH
train:FCVID test:YFCC	-30.0⇓	-2.47⇓	-13.8⇓	-10.0⇓	-5.9⇓
train:YFCC test:FCVID	3.64 ⇑	5.17⇑	-9.70⇓	8.64⇑	2.8⇑

TABLE III: mAP@K results of UVH, UVVH-1 and UVVH-2. For each datasets, the rows above are with 16-bit codes and the ones below are with 128-bit codes.

Datasets	Methods	K=5	K=20	K=40	K=60	K=80
Activity Net	UVH	0.075	0.038	0.024	0.018	0.014
	UVVH-1	0.089	0.043	0.027	0.020	0.016
	UVVH-2	**0.094**	**0.051**	**0.032**	**0.023**	**0.018**
	UVH	0.182	0.088	0.053	0.038	0.030
	UVVH-1	0.187	0.092	0.054	0.039	0.031
	UVVH-2	**0.191**	**0.095**	**0.058**	**0.042**	**0.033**
FCVID	UVH	0.228	0.150	0.118	0.102	0.091
	UVVH-1	0.242	0.158	0.128	0.113	0.103
	UVVH-2	**0.244**	**0.159**	**0.129**	**0.114**	**0.104**
	UVH	0.436	0.252	0.188	0.157	0.136
	UVVH-1	0.457	0.268	0.208	0.179	0.158
	UVVH-2	**0.480**	**0.284**	**0.222**	**0.190**	**0.169**

TABLE IV: mAP@K results of unseen classes retrieval with 64-bit codes.

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
UVH	0.279	0.156	0.098	0.071
UVVH	**0.289**	**0.162**	**0.101**	**0.074**

professional swimming videos.

Effect of the variational mechanism: We show the effect of variational mechanism by comparing UVVH with a fully deterministic method. For the comparison method, we train the 1D-CNN-LSTM model in a fully deterministic way. That is, we set $\boldsymbol{z} = \boldsymbol{\mu}$ and $\alpha_1 = 0$ during training. We use UVH to denote the comparison method. We also train the UVVH model in an end-to-end way, which means the first stage is discarded during training process. We denote it as UVVH-1, and denote the one trained in two stages as UVVH-2. We list the mAP@K results of them in TABLE III. From this table we can see that UVVH-1 consistently outperforms UVH with both code lengths on two datasets, which can validate the effectiveness of variational mechanism. Besides, UVVH-2 slightly outperforms UVVH-1. This shows that using the network pre-trained in deterministic manner, which alleviates the randomness introduced by the monte Carlo sampling procedure in stage2, further improves the performance.

To further validate that the variational mechanism is good for out-of-sample extension, we follow [79] to split FCVID into two parts with no class overlap: train75 and train25/test25, where train75 is the training set and train25/test25 is the retrieval database/query set. The train25/test25 contains videos in 40 categories which are randomly chosen, and the train75 consists of data in the remaining categories. Test25 consists of 1000 query videos and train25 consists of remaining ones. The mAP@K results are shown in TABLE IV. From this table we can see that UVVH outperforms UVH, which validates the effectiveness of variational mechanism for out-of-sample extension. Besides, UVVH outperforms state-of-the-art competitors, SSTH and JTAE, when retrieving data in unseen classes.

Effect of the 1D-CNN-LSTM model: To show the advantage of the 1D-CNN-LSTM model, we compare UVVH with following baselines:

TABLE V: Comparisons with three baselines on FCVID. The rows above are with 16-bit codes and the ones below are with 32-bit codes.

Methods	K=5	K=20	K=40	K=60	K=80	K=100
BL1	0.206	0.103	0.084	0.073	0.065	0.060
BL2	0.160	0.088	0.069	0.058	0.048	0.043
BL3	0.243	0.152	0.121	0.105	0.094	0.086
UVVH	**0.255**	**0.170**	**0.139**	**0.123**	**0.110**	**0.100**
BL1	0.381	0.198	0.150	0.129	0.115	0.105
BL2	0.240	0.149	0.120	0.110	0.102	0.096
BL3	0.394	0.202	0.152	0.126	0.111	0.100
UVVH	**0.411**	**0.226**	**0.174**	**0.149**	**0.133**	**0.120**

- BL1 We extend variational image hashing [63] into video hashing. Specifically, we follow [63] to learn a relaxed binary representation for each frame. We then conduct mean pooling over these representations and binarize the obtained representation to a binary code.

- BL2 We substitute the variational 1D-CNN-LSTM model with a LSTM-based variational autoencoder. LSTM-based encoder-decoder structure is the most widely used framework of existing unsupervised deep video hashing methods.

- BL3 We only use the 1D-deconvolution network to reconstruct the input frame features and rewrite the reconstruction loss L_{recon} as:

$$L_{recon} = \frac{1}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} ||\boldsymbol{v}_i^j - \hat{\boldsymbol{v}}_i^j||_2^2, \tag{17}$$

where $\hat{\boldsymbol{v}}_i^j$ is the j-th intermediate frame feature of the i-th video generated by the 1D-deconvolution network.

The mAP@K results of baselines introduced above and UVVH are shown in TABLE V. In terms of mAP@5, UVVH outperforms BL1 by 18.0% and 3.4% with 16 bits and 32 bits respectively. This indicates that directly extending variational image hashing to variational video hashing will cause performance degradation. A main reason for the degradation is that high-level structured content across frames is neglected. In order to achieve better video retrieval, we should design a suitable network to exploit the correlations among frames. As can be seen, BL2 has the worst retrieval performance. While LSTM is widely used for video representation learning, the long process path of LSTM autoencoder makes it unsuitable for video hashing since important information in early inputs is easily forgotten. Besides, BL3 outperforms BL2 by a great margin. This indicates that 1D-CNN we use in this work can alleviate this problem since it processes the input frames in a hierarchical and parallel manner. Moreover, UVVH outperforms BL3, which indicates that cascading the LSTM decoder with 1D-deconvolution decoder further brings improvement. We owe the performance superiority to our 1D-CNN-LSTM structure, since it integrates the advantage of 1D-CNN and LSTM, alleviating information forgetting and further capturing the correlations among frames.

Effect of intermediate mapping to the high dimensional vector $\boldsymbol{t}_p$: To validate the effect of the intermediate mapping

TABLE VI: mAP@5 results for UVVH and UVVH-nopro on FCVID.

code length	16 bits	64 bits	128 bits
UVVH-nopro	0.228	0.431	0.459
UVVH	**0.244**	**0.451**	**0.469**

TABLE VII: mAP@K results of UVVH-q and UVVH on FCVID. The rows above are with 16-bit codes and the ones below are with 128-bit codes.

Methods	K=5	K=20	K=40	K=60	K=80	K=100
UVVH-q	0.197	0.107	0.078	0.065	0.057	0.050
UVVH	**0.244**	**0.159**	**0.129**	**0.114**	**0.104**	**0.056**
UVVH-q	0.432	0.267	0.209	0.179	0.160	0.145
UVVH	**0.485**	**0.290**	**0.227**	**0.194**	**0.173**	**0.157**

to the high dimensional vector $\boldsymbol{t}_p$ in Equation (11), we directly map $\boldsymbol{h}$ to $\hat{\boldsymbol{t}}$ as the comparison method and denote it as UVVH-nopro. We present the mAP@5 results for UVVH and UVVH-nopro in TABLE VI. As shown in TABLE VI, the intermediate mapping leads to 6.6% improvement for 16-bit codes, 4.6% improvement for 64-bit codes and 2.1% improvement for 128-bit codes. In general the intermediate mapping has positive effect with different code lengths, and short codes benefit more from it.

Effect of the approximate activation function We have different ways to solve the non-convex optimization problem when discretizing the probabilistic latent representation. We use the solution in [31] to handle this problem in this work. Another way to solve it is to add a quantization loss:

$$L_{quan} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{h}_i - \boldsymbol{z}_i||_2. \tag{18}$$

Then the loss function is rewritten as

$$L = \alpha_1 L_{prob} + \alpha_2 L_{recon} + \alpha_3 L_{quan}, \tag{19}$$

where α_1, α_2 and α_3 are hyper-parameters. We name this compared method as UVVH-q and keep the original solution as UVVH. We try some empirical values of these hyper-parameters and include the best mAP@K results of UVVH-q into comparison as shown Table VII. It shows that both method can achieve satisfactory performance. The approximate activation function we use in this paper can achieve slightly better performance without introducing extra hyper-parameter.

V. Conclusion

In this work, we propose an unsupervised variational video hashing method for scalable video retrieval. The framework of UVVH is in a variational encoder-decoder mode. UVVH integrates variational mechanism to learn a probabilistic latent representation of the salient factors of video variations. To better capture the global information, the 1D-CNN-LSTM model encodes the input frame-level features in a parallel and hierarchical way, and further exploits the correlation information in the decoding stage. Extensive experiments on three large-scale video datasets demonstrate the effectiveness of UVVH. For the future work, it is worth to try combining motion features and frame-level appearance features to learn video representation.

References

[1] 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.

[2] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009, vol. 282, no. 3, pp. 1753–1760.

[3] 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.

[4] 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.

[5] 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*, vol. 157, no. 10, 2012, pp. 2074–2081.

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

[7] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, and P. P. K. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1225–1235, 2015.

[8] Y. Jiang, J. Wang, X. Xue, and S. Chang, "Query-adaptive image search with hash codes," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 442–453, 2013.

[9] J. Lu, J. Hu, and Y.-P. Tan, "Discriminative deep metric learning for face and kinship verification," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4269–4282, 2017.

[10] 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.

[11] 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.

[12] 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.

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

[14] Y. Hao, T. Mu, R. Hong, M. Wang, N. An, and J. Y. Goulermas, "Stochastic multiview hashing for large-scale near-duplicate video retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 1–14, 2017.

[15] Y. Hao, T. Mu, J. 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.

[16] J. Song, Y. Yang, Z. Huang, H. 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.

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

[18] 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.

[19] C.L.Chou, H.T.Chen, and S.Y.Lee, "Pattern-based near-duplicate video retrieval and localization on web-scale videos," *IEEE Transactions on Multimedia*, vol. 17, no. 3, pp. 382–395, 2015.

[20] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *IEEE Transactions on Systems Man & Cybernetics Part C*, vol. 41, no. 6, pp. 797–819, 2011.

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

[22] H. P. Gao and Z. Q. Yang, "Content based video retrieval using spatiotemporal salient objects," in *International Symposium on Intelligence Information Processing and Trusted Computing*, 2010, pp. 689–692.

[23] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 5, no. 2, pp. 157–166, 1994.

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

[25] 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.

[26] J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 753–760.

[27] J. Wang, T. Zhang, J. Song, N. Sebe, and H. Shen, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2018.

[28] V. 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.

[29] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.

[30] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.

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

[32] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *IEEE International Conference on Computer Vision*, 2017, pp. 5609–5618.

[33] Y. Duan, J. Lu, Z. Wang, J. Feng, and J. Zhou, "Learning deep binary descriptor with multi-quantization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1183–1192.

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

[35] G. Wu, J. Han, Z. Lin, G. Ding, B. Zhang, and Q. Ni, "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2018.

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

[37] 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.

[38] J. Sun, H. Qi, J. Li, W. Wan, and Q. Wu, "A 3D-CNN based video hashing method," *Tenth International Conference on Digital Image Processing*, p. 82, 08 2018.

[39] 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 Trans. Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

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

[41] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 9, no. 3, 2015, pp. 85–91.

[42] Z. Guo, L. Gao, J. Song, X. Xu, J. Shao, and H. T. Shen, "Attention-based LSTM with semantic consistency for videos captioning," in *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, 2016, pp. 357–361.

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

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

[45] 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.

[46] 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.

[47] 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.

[48] Z. Gan, Y. Pu, R. Henao, C. Li, X. He, and L. Carin, "Learning generic sentence representations using convolutional neural networks," in *Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2390–2400.

[49] 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.

[50] M. Rochan, L. Ye, and Y. Wang, "Video summarization using fully convolutional sequence networks," *CoRR*, vol. abs/1805.10538, 2018.

[51] D. Guo, W. Li, and X. Fang, "Fully convolutional network for multiscale temporal action proposals," *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, 2018.

[52] D. Jimenez Rezende, S. Mohamed, and D. Wierstra, "Stochastic back-propagation and approximate inference in deep generative models," *International Conference on Machine Learning*, pp. 1278–1286.

[53] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.

[54] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 700–708.

[55] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *CoRR*, vol. abs/1611.02648, 2016.

[56] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 3881–3890.

[57] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: a recurrent neural network for image generation," in *IEEE International Conference on Computer Vision*, 2015, pp. 1462–1471.

[58] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *European Conference on Computer Vision*, 2016, pp. 835–851.

[59] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Neural Information Processing Systems*, 2016, pp. 2352–2360.

[60] V. E. Liong, J. Lu, Y. Tan, and J. Zhou, "Cross-modal deep variational hashing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3270–3278.

[61] S. Chaidaroon and Y. Fang, "Variational deep semantic hashing for text documents," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 75–84.

[62] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *International Conference on Machine Learning*, 2017, pp. 913–922.

[63] Y. Shen, L. Liu, and L. Shao, "Unsupervised binary representation learning with deep variational networks," *International Journal of Computer Vision*, 2019.

[64] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. V. zquez, and A. C. Courville, "PixelVAE: a latent variable model for natural images," in *CoRR abs/1511.06434*, 2016.

[65] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *CoRR abs/1511.06434*, 2015.

[66] L. Kaiser and S. Bengio, "Discrete autoencoders for sequence models," *CoRR*, vol. abs/1801.09797, 2018.

[67] 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.

[68] Y.-G. Jiang, Z. W. J. Wang, X. Xue, and S.-F. Chang, "Exploiting feature and class relationships in video categorization with regularized deep neural networks," vol. 40, no. 2, pp. 352–364, 2018.

[69] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in

IEEE Conference on Computer Vision and Pattern Recognition, pp. 961–970.
[70] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "The new data and new challenges in multimedia research," vol. 59, no. 2, pp. 64–73, 2016.
[71] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.
[72] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quenot, "An overview of the goals tasks data, evaluation mechanisms and metrics," in *Proceedings of TREC Video Retrieval Evaluation*, 2014.
[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[74] 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.
[75] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[76] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Security*, 2010, pp. 249–256.
[77] D. Kingma and J. Ba., "Adam: a method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.
[78] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[79] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?" in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, 2017, pp. 1732–1736.

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.

Zhixiang Chen received the B.S. degree in microelectronics from the Xi'an Jiaotong University, Xi'an, China and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, China. He is now with Department of Electrical and Electronic Engineering of Imperial College London, United Kingdom. His current research interests include face analysis, large scale visual search, 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.