Dual-Stream Knowledge-Preserving Hashing for Unsupervised Video Retrieval (Supplementary Material)

Pandeng Li[1], Hongtao Xie[1], Jiannan Ge[1], Lei Zhang[2], Shaobo Min[3], and Yongdong Zhang[1]

[1] University of Science and Technology of China
[2] Kuaishou Technology
[3] Tencent Data Platform
{lpd,gejn}@mail.ustc.edu.cn, {htxie,zhyd73}@ustc.edu.cn, zhanglei06@kuaishou.com, bobmin@tencent.com

This supplementary material provides more details of the proposed Dual-stream Knowledge-Preserving Hashing (DKPH) framework: (1) Time efficiency of DKPH; (2) More experiment results; (3) Pseudo code of DKPH.

1 Time Efficiency of DKPH

Training time. We train our teacher-student model from scratch on the Pytorch framework [4] with a single TITAN X GPU with 12GB memory. Table 1 shows the training time of DKPH. Compared to BTH [3], DKPH reduces training time by about $1\% \sim 8\%$ on FCVID [2], ActivityNet [1] and YFCC [5]. Since DKPH has a dual-stream structure to deal with the task heterogeneity problem, binary codes can focus on learning semantic knowledge. Therefore, DKPH converges faster than BTH, resulting in a shorter training time. For example, DKPH achieves optimal results at 48 epoch on FCVID, while BTH is at 55 epoch.
Testing encoding time. The time cost to generate binary codes for query videos is crucial to evaluate the practical retrieval system. Table 1 also shows the encoding efficiency of BTH and DKPH. Because the hash layer and the temporal layer in the dual-stream structure are designed in parallel, which has less impact on the encoding time.

2 More Experiment Results

More cross-dataset evaluation comparisons. As the essence of DKPH is to focus on capturing the semantic-dependent discriminative similarity information, DKPH could provide robust binary codes. To demonstrate this, we investigate the generalization of DKPH for cross-dataset retrieval. We train DKPH and BTH [3] on FCVID and test on YFCC in Table 2, which shows MAP@k results for cross-dataset retrieval at different bits. DKPH improves MAP@k performances for all bits. These gains show that the dual-stream structure can additionally provide beneficial information for hashing learning, and establish

Table 1. The training and testing encoding time for BTH and DKPH at 16 bits on FCVID, ActivityNet and YFCC.

Method	Training time			Testing encoding time		
	FCVID	ActivityNet	YFCC	FCVID	ActivityNet	YFCC
BTH [3]	5018.84s	1894.22s	75021.11s	0.294ms	0.351ms	0.309ms
DKPH	4860.41s	1738.59s	74273.74s	0.296ms	0.353ms	0.309ms

Table 2. Cross-dataset MAP@k results when training on FCVID and test on YFCC.

Method	16 bits				32 bits				64 bits			
	k=5	k=20	k=60	k=100	k=5	k=20	k=60	k=100	k=5	k=20	k=60	k=100
BTH [3]	0.151	0.098	0.073	0.065	0.298	0.123	0.099	0.081	0.331	0.134	0.108	0.101
DKPH	**0.189**	**0.109**	**0.089**	**0.077**	**0.312**	**0.153**	**0.104**	**0.089**	**0.342**	**0.171**	**0.119**	**0.106**

the importance of information decomposition in video hashing. Binary codes focus on semantic concepts rather than the underlying reconstruction information, ensuring good transferability and generalization of DKPH when retrieving unknown datasets.

Iterative training. We try to update features of trained $\boldsymbol{\Omega}^S$ to construct the Gaussian-adaptive graph $\hat{\boldsymbol{A}}$, and train again (DKPH-I). However, DKPH-I does not bring significant improvement in Table 3. Considering that iterative training brings more consumption and less benefit, we finally do not adopt this strategy.

Effects of loss weights. We further study the influence of loss weights γ_1 and γ_2 . Fig 1 shows the effect of γ_1 on FCVID dataset. We fix γ_2 to 0.9 and evaluate the MAP@k by varying γ_1 between 0.05 and 0.17. The performance shows that our model performs relatively better with γ_1 between 0.08 and 0.14.

Fig 2 shows the effect of γ_2 on FCVID dataset. We fix γ_1 to 0.11 and evaluate the MAP@k by varying γ_2 between 0.5 and 1.3. Experimental results show that our DKPH is quite robust to this parameter. For values from 0.5 to 1.3, the trained models consistently achieve promising results. We attribute these achievements to such a guess that $\mathcal{L}_{tsim}$ and $\mathcal{L}_{bsim}$ are complementary losses. $\mathcal{L}_{bsim}$ acts on binary representations directly, while $\mathcal{L}_{tsim}$ focuses on how to align the teacher-student visual embeddings to learn the semantic knowledge of the teacher model. $\mathcal{L}_{tsim}$ only implicitly affects binary codes, so adjustments to the weight γ_2 have the slight effect on the results.

Qualitative results. Top-5 retrieval results are illustrated in Fig. 3. It is difficult for SSVH to distinguish different semantics, our model can still search videos in the same class, which proves the discrimination of binary codes with the knowledge preservation strategy.

3 Pseudo Code of DKPH

Pseudo code of the student model. Algorithm 1 provides the pseudo-code of the forward process of the student model in a PyTorch-like style. The student

Table 3. Iterative training on FCVID.

Method	16 bits				32 bits				64 bits			
	k=5	k=20	k=60	k=100	k=5	k=20	k=60	k=100	k=5	k=20	k=60	k=100
DKPH	**0.297**	**0.174**	0.120	0.097	**0.441**	0.275	0.203	0.171	**0.494**	0.331	0.255	0.228
DKPH-I	0.293	0.172	**0.121**	**0.098**	0.439	**0.277**	**0.205**	**0.172**	0.491	**0.332**	**0.257**	**0.229**

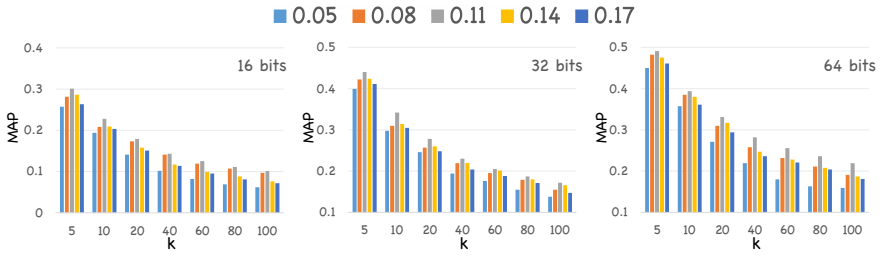

Fig. 1. The MAP@k scores with various configurations about the weight γ_1 of binary structure similarity loss $\mathcal{L}_{bsim}$ on FCVID.

model designs a simple but effective dual-stream structure, containing a temporal layer and a hash layer. The goal of dual-stream structure: the temporal layer tries to capture reconstruction-dependent information by learning dynamic frame-level features, while the hash layer focuses on the semantic-dependent part from a global video-level perspective.

Pseudo code of the training process. The step-by-step training description of the proposed DKPH is summarized in Algorithm 2.

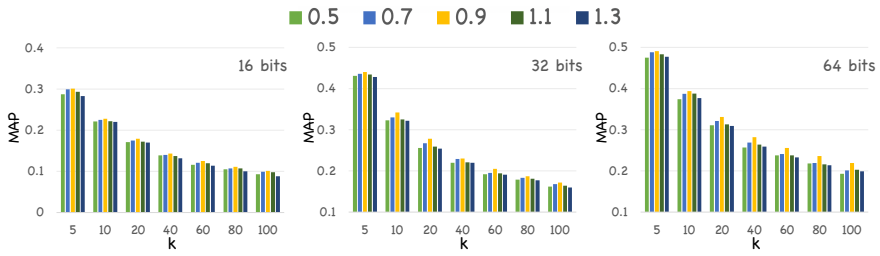

Fig. 2. The MAP@k scores with various configurations about the weight γ_2 of visual embedding similarity loss $\mathcal{L}_{tsim}$ on FCVID.

Fig. 3. Top-5 retrieval results on FCVID and ActivityNet at 16 bits. Positives are highlighted with green, while negatives are red.

Algorithm 1. Pseudo code of the student model in a PyTorch-like style.

```
# cat: concatenation
def forward(self, x): # input frame features: B × M × D
    # transformer encoder
    t = self.transformer(x)# visual embeddings : B × M × d

    # temporal layer
    l = self.temporal(t)# latent features : B × M × K

    # hash layer
    hid_b = self.hash(t.reshape([B, M × d]))# real-valued codes: B × K
    # binarization
    b = self.binary_tanh(hid_b)# binary codes: B × K

    # decoder
    x̃ = self.decoder(cat(torch.unsqueeze(b,dim=1).repeat(1,M,1),l))#
    decoder features : B × M × D
return b, x̃, t
```

Algorithm 2. The Training Process of DKPH.

Input: Unlabeled video points $\mathcal{V} = \{v_i\}_{i=1}^{N} \in \mathbb{R}^{N\times M\times D}$; Teacher epochs Γ^T; Student epochs Γ^S; Batch size B; Teacher code length K^T; Student code length K;
Output: The trained student network Ω^S;

Initialize: Randomly initialize the teacher model Ω^T and the student model Ω^S;
Warm up:
for i in $0, \ldots, \Gamma^T$ **do**
  for j in $0, \ldots, \lfloor \frac{N}{B} \rfloor$ **do**
    Randomly mask batch video points $\mathcal{V}_j^B$;
    Visual embeddings $\mathcal{T}_j \in \mathbb{R}^{B\times M\times d}$, Binary codes $\mathcal{B}_j \in \{-1,1\}^{B\times M\times K^T}$, Decoder video points $\tilde{\mathcal{V}}_j \in \mathbb{R}^{B\times M\times D} \leftarrow \Omega^T\left(\mathcal{V}_j^B\right)$;
    Calculate $\mathcal{L}_{recon}$ with $\mathcal{V}_j^B, \tilde{\mathcal{V}}_j$;
    Update the teacher model parameters by the backpropagation algorithm;
  end for
end for
Graph construction:
Calculate teacher visual embeddings $\mathcal{T} \in \mathbb{R}^{N\times M\times d}$ with the trained Ω^T;
Clustering center $\mathcal{C} \in \mathbb{R}^{N_c\times d} \leftarrow$ K-means clustering for video-level teacher visual embeddings $\bar{\mathcal{T}} \in \mathbb{R}^{N\times d}$;
Calculate 1-NN nearest center $\mathcal{C}_1 \in \mathbb{R}^{N\times d}$ of corresponding video-level visual embedding $\bar{\mathcal{T}}$ with $\mathcal{C}$;
Calculate the approximate graph adjacency $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ with $\bar{\mathcal{T}}$, $\mathcal{C}$;
Calculate the Gaussian-adaptive graph adjacency matrix $\hat{\boldsymbol{A}} \in \{-1,0,1\}^{N\times N}$;
Knowledge preservation:
for i in $0, \ldots, \Gamma^S$ **do**
  for j in $0, \ldots, \lfloor \frac{N}{B} \rfloor$ **do**
    Equal sampling positive and negative points $\mathcal{V}_{j'}^B$ for $\mathcal{V}_j^B$ with $\hat{\boldsymbol{A}}$;
    Randomly mask batch video points $\mathcal{V}_j^B$, $\mathcal{V}_{j'}^B$;
    Visual embeddings $\mathcal{T}_j \in \mathbb{R}^{B\times M\times d}$, Binary codes $\mathcal{B}_j \in \{-1,1\}^{B\times K}$, Decoder video points $\tilde{\mathcal{V}}_j \in \mathbb{R}^{B\times M\times D} \leftarrow \Omega^S\left(\mathcal{V}_j^B\right)$;
    Visual embeddings $\mathcal{T}_{j'} \in \mathbb{R}^{B\times M\times d}$, Binary codes $\mathcal{B}_{j'} \in \{-1,1\}^{B\times K}$, Decoder video points $\tilde{\mathcal{V}}_{j'} \in \mathbb{R}^{B\times M\times D} \leftarrow \Omega^S\left(\mathcal{V}_{j'}^B\right)$;
    Calculate $\mathcal{L}_{recon}$ with $\mathcal{V}_j^B, \tilde{\mathcal{V}}_j, \mathcal{V}_{j'}^B, \tilde{\mathcal{V}}_{j'}$;
    Calculate $\mathcal{L}_{bsim}$ with $\mathcal{B}_j, \mathcal{B}_{j'}, \hat{\boldsymbol{A}}$;
    Calculate $\mathcal{L}_{tsim}$ with $\mathcal{T}_j, \mathcal{T}_{j'}, \hat{\boldsymbol{A}}, \mathcal{C}_1$;
    Update the student model parameters by the backpropagation algorithm;
  end for
end for

References

1. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: CVPR (2015)
2. Jiang, Y.G., Wu, Z., Wang, J., Xue, X., Chang, S.F.: Exploiting feature and class relationships in video categorization with regularized deep neural networks. TPAMI (2017)
3. Li, S., Li, X., Lu, J., Zhou, J.: Self-supervised video hashing via bidirectional transformers. In: CVPR (2021)
4. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS (2019)
5. Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: The new data and new challenges in multimedia research. arXiv preprint arXiv:1503.01817 (2015)