Context Encoding for Video Retrieval with Contrastive Learning

Jie Shao∗
shaojie@fudan.edu.cn
Fudan University
ByteDance AI Lab

Xin Wen†
wxn9@tongji.edu.cn
Tongji University
ByteDance AI Lab

Bingchen Zhao
zhaobc@tongji.edu.cn
Tongji University

Changhu Wang
wangchanghu@bytedance.com
ByteDance AI Lab

Xiangyang Xue
xyxue@fudan.edu.cn
Fudan University

ABSTRACT
Content-based video retrieval plays an important role in areas such as video recommendation, copyright protection, etc. Existing video retrieval methods mainly extract frame-level features independently, therefore lack of efficient aggregation of features between frames, and it is difficult to effectively deal with poor quality frames, such as frames with motion blur, out of focus, etc. In this paper, we propose CECL (Context Encoding for video retrieval with Contrastive Learning), a video representation learning framework that aggregates the context information of frame-level descriptors, and a supervised contrastive learning method that performs automatic hard negative mining, and utilizes the memory bank mechanism to increase the capacity of negative samples. Extensive experiments are conducted on multi video retrieval tasks, such as FIVR, CC_WEB_VIDEO and EVVE. The proposed method shows a significant performance advantage (~17% mAP on FIVR-200K) over state-of-the-art methods with video-level features, and deliver competitive results with a much lower computational cost when compared with frame-level features.

CCS CONCEPTS
• Information systems → Query representation; Retrieval models and ranking; Near-duplicate and plagiarism detection; Clustering and classification; • Computing methodologies → Visual content-based indexing and retrieval.

KEYWORDS
Representation Learning, Video Retrieval, Contrastive Learning

1 INTRODUCTION
Content-based video retrieval is critical for applications like video recommendation, copyright protection, etc. The content on the Internet has evolved from the previous plain text to various forms of multimedia presentation, such as pictures, audio, and video. In particular, the rapid growth of various video content (such as long video, short video, live broadcast, etc.) has brought huge challenges to video retrieval methods.

Video retrieval approaches mainly follow the scheme of calculating the similarity between videos based on video-level representations or frame-level representations. For those based on video-level representations, code books [6, 35, 38] or hashing functions [59, 60] was employed in the early studies, and later Deep Metric Learning (DML) was also used to train a network with triplet loss to learn a better video-level representation [36]. Other approaches typically extract frame-level representations to apply frame-to-frame similarity measurement and then aggregate them into video-level similarities [10, 34, 40, 62]. With more elaborate similarity measures, they typically outperform those based on video-level representations. Recently ViSiL [34] trained a subnet to refine the video-level similarity matrix for similarity measurement and reached state-of-the-art performance in several video retrieval tasks, however, the computational cost is heavy.

With video-level representations or frame-level representations, the above approaches lead to two research focus: to learn a better representation or a better similarity measure. Although the latter approach reaches better performance, we argue that a more versatile and efficient way should be to optimize the video representation rather than the similarity measure. Another issue is that for both approaches, the initial frame-level representations are extracted independently as image representations. However, in contrast to images, frames extracted from videos often suffer from motion blur, occlusion and out of focus, and such inferior frames usually convey less information. A natural idea is to exploit the context information in the temporal axis. Considering spatio-temporal video representation and matching, methods based on Recurrent Neural Networks...
Convolutions (R-MAC) [64] build feature vectors that encode several image regions rather than the whole image, and Ly-iMAC [34] applies R-MAC on the activations of the intermediate convolutional layers, but the regional feature maps are stacked rather than summed. Besides variants of MAC, Sum-Pooled Convolutional features (SPoC) [3] and Generalized Mean (GeM) [16] pooling are also considerable counterparts.

### 2.2 Feature Aggregation

Typically, the video feature aggregation paradigm can be divided into two categories: (1) local feature aggregation models [11, 27, 48, 57] which are derived from traditional local image feature aggregation models, and (2) sequence models [9, 12, 14, 22, 65, 75] that model the temporal order of the video representation.

The commonly used local feature aggregation models include Bag-of-Words [11, 57], Fisher Vector [48], and Vector of Locally Aggregated Descriptors (VLAD) [27], of which the unsupervised learning of a visual code book is required. The NetVLAD [1] transfers VLAD into a differential version, and the clusters are tuned via back-propagation instead of k-means clustering. NeXtVLAD [39] further decomposes the high-dimensional feature into a group of relatively low-dimensional vectors with attention before applying NetVLAD aggregation over time, which is both effective and parameter efficient. In terms of the sequence models, the Long Short-Term Memory (LSTM) [22] and Gated Recurrent Unit (GRU) [9] are commonly used to model contextual information within a long-range for video re-localization and copy detection [14, 24]. Besides, the effectiveness of self-attention in capturing short and long-range dependency with attention mechanism has been proved with the success of Transformer [65]. For the feature aggregation of videos, this also shows success in video classification [67] and object detection [23], opening new possibilities for feature aggregation for video retrieval.

### 2.3 Metric Learning

Metric learning aims to learn an embedding that minimizes the distance between related samples and maximizes it between irrelevant ones. Metric learning has been commonly used in face recognition [7, 53, 70], image retrieval [45, 58, 68, 71] and video retrieval [34, 36]. With only pair-wise labels available, the triplet loss [69] is commonly used in video retrieval tasks [34, 36]. The classic approach in [36] performs hard negative mining to generate hard triplets, but despite both the off-line triplet generation stage and the training stage are time-consuming, the information that triplets can convey is limited [58]. Although [20] showed the triplet loss can perform competitively against other popular metric learning approaches with proper hard negative sampling strategy, the proposed PK sampling strategy is only compatible with datasets with class-level labels.

Contrastive learning has become the common training architecture of recent self-supervised learning works [8, 18, 21, 46, 63], in which the positive and negative sample pairs are constructed with a pretext task in advance, and the model tries to distinguish the positive sample from massive randomly sampled negative samples in a classification manner. The contrastive loss typically performs better in general than triplet loss on representation tasks [8], as
the triplet loss can only handle one positive and negative at a time. The core of the effectiveness of contrastive learning is the use of rich negative samples [63], one approach is to sample them from a shared memory bank [74], and [18] replaced the bank with a queue and used a moving-averaged encoder to build a larger and consistent dictionary on-the-fly. Apart from self-supervised learning, supervised contrastive learning for classification tasks is also discussed in [31], in which a modified batch contrastive loss that supports an arbitrary number of positives is proposed to leverage label information effectively. As we only have pair-wise labels, our supervised contrastive learning approach is more similar to the self-supervised approach, where each anchor is coupled with only one positive.

3 METHOD

In this section, we inArst formally deInAne the video representation learning problem (Section 3.1) and describe the frame-level feature extraction step (Section 3.2). Then, we demonstrate the joint-feature aggregation approach (Section 3.3) and the contrastive learning method based on pair-wise video labels (Section 3.4), then conduct further analysis on the gradients of the loss function (Section 3.5). And last, we discuss the similarity measure of aggregated video-level and frame-level video descriptors (Section 3.6).

3.1 Problem Setting

Video representation learning is a task of learning an embedding function \( f(\cdot) \) that transforms the original video descriptor \( x \) into another representation \( f(x) \), which is easier to extract useful information for downstream tasks. As we only consider the RGB data of a video, each video representation \( x \) can be raw pixels \( (x \in \mathbb{R}^{m \times n \times f}, \) where each video contains \( f \) frames and each frame is \( m \times n \) dimensional, or some frame-level descriptors \( (x \in \mathbb{R}^{d \times f}, \) where \( d \) is the dimensionality of the frame-level feature) in which each frame is encoded separately and then stacked together, or a compact video-level descriptor \( x \in \mathbb{R}^{d}, \) where \( d \) is the dimensionality of the video-level feature).

We address the problem of video representation learning for Near-Duplicate Video Retrieval (NDVR), Fine-grained Incident Video Retrieval (FIVR), and Event Video Retrieval (EVR) tasks. For all three tasks, there are no explicit classes as the content of a single video can be complicated, making it hard to apply popular classification-based video representation learning models. What we have are pair-wise labels describing whether two videos are similar (near duplicate, complementary scene, same event, etc.) or not (distractors). Given such pair-wise labels, metric learning can be a good way to tackle.

We view metric learning from a similarity optimization perspective. Take the similarity function as \( \text{sim}(\cdot, \cdot) \), the similarity of two video descriptor \( x, y \) can be denoted as \( \text{sim}(x, y) \). Given these, our task is to optimize the embedding function \( f(\cdot) \), such that \( \text{sim}(f(x), f(y)) \) is maximized if \( x \) and \( y \) are similar videos, and minimized otherwise. The similarity function is typically euclidean similarity or cosine similarity, but can be any other function within range \([0, 1]\) or \([-1, 1]\). The embedding function \( f(\cdot) \) typically takes a video-level descriptor \( x \in \mathbb{R}^{d} \) and returns an embedding \( f(x) \in \mathbb{R}^{k} \), where \( k \ll d \). However, in our work, \( f(\cdot) \) is a feature aggregation model, thus frame-level descriptors \( x \in \mathbb{R}^{d \times f} \) are taken as input, and the output can be both aggregated video-level descriptor \( f(x) \in \mathbb{R}^{d} \) and refined frame-level descriptors \( f(f(x)) \in \mathbb{R}^{d \times f} \).

3.2 Feature Extraction

Here we consider the frame-level feature extraction process. According to the results reported in [34](Table 2), we select iMAC [15] and L3-iMAC [34] as our benchmark frame-level feature extraction methods. Given a pre-trained CNN network with \( K \) convolutional layers, \( K \) feature maps \( M^{k} \in \mathbb{R}^{n_{k}^{x} \times n_{k}^{y} \times c_{k}} \) \((k = 1, \ldots, K)\) are generated, where \( n_{k}^{x} \times n_{k}^{y} \) is the dimension of each feature map of the \( k \)th layer, and \( c_{k} \) is the total number of channels.

For iMAC feature, the maximum value of every channel of each layer is extracted to generate \( K \) feature maps \( M^{k} \in \mathbb{R}^{c_{k}}, \) as formulated in Eq. 1:

\[
\psi^{k}(i) = \max M^{k}(\cdot, \cdot, i), \quad i = 1, 2, \ldots, c_{k},
\]

where layer vector \( \psi^{k} \) is a \( c_{k} \)-dimensional vector that is derived from max pooling on every channel of feature map \( M^{k} \).

For L3-iMAC feature, max pooling with different kernel size and stride are applied to every channel of different layers to generate \( K \) feature maps \( M^{k} \in \mathbb{R}^{3 \times 3 \times c_{k}} \). Unlike the setting of L3-iMAC, we then follow the tradition of R-MAC to sum the \( 3 \times 3 \) feature maps together, then apply \( L_{2} \)-normalization on each channel to form a feature map \( M^{k} \in \mathbb{R}^{c_{k}} \). This approach keeps the dimensionality low which is equal to the iMAC feature, we denote this approach as L3-iRMAC. This presents a trade-off between the preservation of fine-trained spatial information and low feature dimensionality.

For both iMAC and L3-iRMAC, all layer vectors are concatenated to a single descriptor after extraction, then PCA is applied to perform whitening and dimensionality reduction following the common practice [26, 34], finally \( L_{2} \)-normalization is applied on each channel, resulting in a compact frame-level descriptor. By applying this process to each extracted frames of a video, we get the frame-level video descriptor \( x \in \mathbb{R}^{d \times f} \).

3.3 Feature Aggregation

In this section, we discuss the details about the feature aggregation model/function \( f(\cdot) \). After performing feature extraction, a sequence of frame-level descriptors \( x \in \mathbb{R}^{d \times f} \) of a video is obtained. This is then passed to the feature aggregation model \( f(\cdot) \) to generate a video-level descriptor \( f(x) \in \mathbb{R}^{d} \), or frame-level descriptors \( f(f(x)) \in \mathbb{R}^{d \times f} \), as illustrated in Figure 2.

Local feature aggregation models. For the NetVLAD [1] and the NeXtVLAD [39] model, when applied to the aggregation of frame-level video descriptors, each descriptor is treated as a local image descriptor as [44]. Following the setting of [39, 44], the Context Gating module is used in both models. As the local feature aggregation models do not model the temporal order, we only use them for aggregating compact video descriptors \( f(x) \in \mathbb{R}^{K} \).

Sequence aggregation models. Typically, a sequence model takes the input sequence one at a time, generating a sequence of hidden states \( h_{t} \) as a function of the previous hidden state \( h_{t-1} \) and the
current input at position \( t \). Denote the hidden state at the \( t \)-th time step as \( h_t \), the encoding process of the LSTM [22] and GRU [9] can be written as:

\[
\begin{align*}
    h_t^L &= \text{LSTM} \left( x_t, h_{t-1} \right), \\
    h_t^G &= \text{GRU} \left( x_t, h_{t-1} \right),
\end{align*}
\]

(2)

respectively. Due to natural characteristics and behaviors of recurrent models (LSTM and GRU), the hidden states can encode and aggregate the previous contextual information. By concatenating all the yielded hidden states following time order, we get the aggregated video representation:

\[
    f_{\text{recurrent}}(x) = [h_0, \ldots, h_{T-1}].
\]

(3)

For the Transformer [65] model, following the setting of [14, 75], only the encoder structure of the sequence models is used. With the parameter matrices written as \( W_Q, W_K, W_V \), the entire video descriptor \( x \in \mathbb{R}^{d \times f} \) is first encoded into Query \( Q \), Key \( K \) and Value \( V \) by three different linear transformations: \( Q = x^T W_Q, K = x^T W_K \) and \( V = x^T W_V \). This is further calculated by the self-attention layer as:

\[
    \text{Attention}(Q,K,V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) V.
\]

(4)

The result is then taken to the LayerNorm layer [2] and Feed Forward Layer [65] to get the output of the Transformer encoder, i.e. \( f_{\text{transformer}}(x) \in \mathbb{R}^{d \times f} \). The multi-head attention mechanism is also used in our implementation. Although the encoded feature keep the same shape as the input, the contextual information within a longer range of each frame-level descriptor is incorporated. Besides, for both recurrent models (LSTM and GRU) and the Transformer model, by simply averaging the encoded frame-level video descriptors along the time axis, we can also get the compact video-level representation \( \bar{f}(x) \in \mathbb{R}^d \).

### 3.4 Contrastive Learning

If we denote \( w_a, w_p, w_n(j = 1, 2, \ldots, N-1) \) as the video-level representations before applying normalization of the anchor, positive, negative examples, we get the similarity scores by:

\[
    s_p = w_a^T w_p / \left( \|w_a\| \|w_p\| \right) \quad \text{and} \quad s_n = w_a^T w_n / \left( \|w_a\| \|w_n\| \right).
\]

Then the InfoNCE [46] loss is thus written as:

\[
\mathcal{L}_{\text{nce}} = - \log \frac{\exp(s_p / \tau)}{\exp(s_p) + \sum_{j=1}^{N-1} \exp(s_n / \tau)},
\]

(5)

where \( \tau \) is a temperature hyper-parameter [74]. To utilize more negative samples for better performance, we borrow the idea of the memory bank in [74]. For each batch, we take one positive pair from the core dataset and randomly sample \( n \) negative samples from distractors, then the compact video-level descriptors are generated with a shared encoder. The negative samples of all batches and all GPUs are concatenated together to form the memory bank. We compare the similarity of the anchor sample against the positive sample and all negatives in the memory bank, resulting in \( s_{p} \) and \( s_{n} \). Then the loss can be calculated in a classification manner following Eq. 5 and 6.

\[
\mathcal{L}_{\text{circle}} = - \log \frac{\exp(r \alpha \gamma (s_p - \Delta_p))}{\exp(r \alpha \gamma (s_p - \Delta_p)) + \sum_{j=1}^{N-1} \exp(r \alpha \gamma (s_n - \Delta_n))},
\]

(6)
where $γ$ is the scale factor (equivalent with the parameter $τ$ in Eq. 5), and $m$ is the relaxation margin. $α_p = [1 + m - sp]_+\cdot α_p^* = \left[\frac{1}{n} + m\right]_+, \Delta_p = 1 - m, \Delta_n = m$. Compared with the InfoNCE loss, the Circle loss optimizes $sp$ and $sn$ separately with adaptive penalty strength and adds within-class and between-class margins.

### 3.5 One Step Further on the Gradients

In the recent work of Khosla et al. [31], the proposed batch contrastive loss is proved to focus on the hard positives and negatives automatically with the help of feature normalization by conducting gradient analysis. We further reveal that this is the common property of Softmax loss and its variants when combined with feature normalization. For simplicity, we analyze the gradients of Softmax loss, the origin of both InfoNCE loss and Circle loss:

$$L_{\text{softmax}} = -\frac{1}{n} \sum_i \log \frac{\exp(s_i)}{\sum_j \exp(s_j)}. \quad (7)$$



the notation is as aforementioned. Here we show that easy negatives contribute the gradient weakly while hard negatives contribute greatly. With the notations declared in Section 3.4, we denote the normalized video-level representation as $z_v = w_v/\|w_v\|$, then the gradients of Eq. 7 with respect to $w_a$ is:

$$\frac{\partial L_{\text{softmax}}}{\partial w_a} = \frac{\partial L_{\text{softmax}}}{\partial a} \cdot \frac{\partial a}{\partial z_a} \approx \frac{1}{\|z_a\|} \left(1 - z_a^T z_a\right) \cdot \left(\sigma(s_p) - 1\right) z_p + \sum_{j=1}^{N} \sigma(s_j) z_j^T z_a,$$

positive

$$\approx \left(1 - \sigma(s_p)\right) \left[(z_a^T z_p) z_a - z_p\right] + \sum_{j=1}^{N} \sigma(s_j) \left(z_a^T z_j^T z_a\right),$$

and the symmetric version:

$$\sim_{\text{sym}}(x, y) = \frac{\left(\sim_{\text{sym}}(x, y) + \sim_{\text{sym}}(y, x)\right)}{2}. \quad (11)$$

Note that this approach (chamfer similarity) seems to be inconsistent with the training target (cosine similarity), where the frame-level video descriptors are averaged into a compact representation and the similarity is calculated with dot product. However, the similarity calculation process of the compact video descriptors can be written as:

$$\sim_{\text{cos}}(x, y) = \frac{1}{n} \sum_{i=0}^{n-1} x_i y_j^T \approx \frac{1}{n} \sum_{i=0}^{n-1} m_{j=0}^{m-1} x_i y_j^T. \quad (12)$$

Therefore, given frame-level features, chamfer similarity averages the maximum value of each row of the video-video similarity matrix, while cosine similarity averages the mean value of each row. It is obvious that $\sim_{\text{cos}}(x, y) \leq \sim_{\text{cos}}(x, y)$ hold true, therefore by optimizing the cosine similarity, we are optimizing the lower-bound of the chamfer similarity. As only the compact video-level feature is required, both time and space complexity are greatly reduced as cosine similarity is much computational efficient.

### 4 EXPERIMENTS

#### 4.1 Experiment Setting

We evaluate the proposed approach on three video retrieval tasks, namely Near-Duplicate Video Retrieval (NDVR), Fine-grained Incident Video Retrieval (IFVR), and Event Video Retrieval (EVR). In all cases, we report the mean Average Precision (mAP).

**Training dataset.** We leverage the VCDB [28] dataset and a subset of the FIVR-200K [33] dataset as training dataset. The core dataset of VCDB has 528 query videos and 6,139 positive pairs, and the distractor dataset has 100,000 distractor videos, of which we combined with feature normalization to perform hard negative mining automatically, and use the memory bank mechanism to increase the capacity of negative samples, which greatly improves the training efficiency and effect.
successfully downloaded 99,181 of them. The FIVR-200K dataset includes 225,960 videos and 100 queries, we successfully downloaded 225,950 of them. Three different fine-grained video retrieval tasks: (1) Duplicate Scene Video Retrieval, (2) Complementary Scene Video Retrieval and (3) Incident Scene Video Retrieval. Following the setting in [33], we order the videos based on their publication time and then split them in half, resulting in the former FIVR-TRAIN dataset with 31 queries and the latter FIVR-TEST dataset with 69 queries. We extract all 6,217 positive pairs in the ISVR task with 69 queries. We extract all 6,217 positive pairs in the ISVR task. For models trained on the FIVR-TRAIN dataset as in [34] is used.

Evaluation dataset. For models trained on the VCDB dataset, we test them on the CC_WEB_VIDEO [72] dataset for NDVR task, FIVR-200K for FIVR task and EVVE [52] for EVR task, for the models trained on the FIVR-TRAIN dataset, we test them on the FIVR-TEST dataset. The CC_WEB_VIDEO dataset contains 24 query videos and 13,129 labeled videos, we managed to download 13,099 of them. The EVVE dataset consists of 2,375 videos and 620 queries, we successfully downloaded the whole dataset.

Implementation Details. For feature extraction, we extract one frame per second for all videos. For all retrieval tasks, we extract the frame-level features following the scheme in Section 3.2. The intermediate features are all extracted from the output of four residual blocks of ResNet-50 [19]. PCA trained on 997,090 randomly sampled frame-level descriptors from VCDB is applied to both iMAC and L3-iRMAC features to perform whitening and reduce its dimension from 3840 to 1024. Finally ℓ2-normalization is applied.

For both NetVLAD and NeXtVLAD, the number of clusters is set to 256, context gating mechanism is used with gating_reduction=8, one fully connected layer is used to reduce the dimension of the final flattened representation to 1024, and a dropout layer with drop_rate=0.5 is applied before the fully connected layer. The expansion ratio of NeXtVLAD is set to 2. For both LSTM and GRU, the number of hidden units is set to 1024, the number of layers set to 2 and dropout_rate set to 0.2. For all these four models, batch normalization [25] is applied before each non-linear layer. For the Transformer, it is implemented with 1 single layer, 8 attention heads, dropout_rate set to 0.5, and the dimension of the feed forward layer set to 2048. No batch normalization is used in the Transformer model as it may speed up over-fitting in practice, interestingly, all other four models won’t converge without it. For both InfoNCE loss and Circle loss, the parameters are set as default: τ = 0.07, γ = 256, m = 0.25. During training, all videos are padded to 300 frames (if longer, a random segment with a length of 300 is extracted), and the full video is used in the evaluation stage. We use Adam [32] as our optimizer, the initial learning rate is set to 10^{-6} for NetVLAD and NeXtVLAD, and 10^{-5} for sequence models, and cosine annealing learning rate scheduler [41] is used. All models are trained with batch size 64, and 16 × 64 negative samples sampled from the distractors are sent to the memory bank each batch, with a single device with 4 Tesla-V100-SXM2-32GB GPUs, the size of the memory bank is equal to 4096. The training of all models stops when over-fitting is observed, i.e. 5, 5, 20, 30, 40 epochs for NetVLAD, NeXtVLAD, LSTM, GRU, and Transformer respectively. All models are implemented with PyTorch [47], and distributed training is implemented with Horovod [55].

4.2 Feature Aggregation Model Comparison

| Model       | CC_WEB_VIDEO2 | FIVR-200K |
|-------------|---------------|-----------|
|             | cc_web        | cc_web*    | DSVR | CSVR | ISVR |
| NetVLAD     | 0.971         | 0.944      | 0.513 | 0.494 | 0.412 |
| NeXtVLAD    | 0.967         | 0.935      | 0.495 | 0.471 | 0.389 |
| LSTM        | 0.969         | 0.937      | 0.505 | 0.483 | 0.400 |
| GRU         | 0.969         | 0.940      | 0.515 | 0.495 | 0.415 |
| Transformer | 0.972         | 0.943      | 0.551 | 0.532 | 0.454 |

This section presents a comparison of the five feature aggregation models. All models are trained on VCDB dataset with iMAC feature to generate compact video-level descriptor, and dot product is used for similarity calculation for both train and evaluation. Table 1 presents the results of the comparison on both CC_WEB_VIDEO and FIVR-200K. As in [44], NetVLAD outperforms the classic sequence models (LSTM, GRU), but interestingly the NeXtVLAD show the worst performance. Besides, the Transformer model demonstrate excellent performance in almost all tasks, indicating that with the spatio-temporal information fully utilized, there are huge potential for the aggregation model to improve. We also present comparison between feature extraction methods and loss functions in Table 2, with loss function fixed to Circle loss. L3-iRMAC show consistent improvement against iMAC, indicating that the local spatial information are leveraged by the L3-iRMAC feature with lower dimensionality maintained. For the loss functions, the InfoNCE loss show notable inferiority compared with Circle with default parameters τ = 0.07, γ = 256, m = 0.25, with temperature parameter τ set to 1/256 (equivalent with γ = 256 in Circle loss), it still show around 0.005 less mAP. Next, we only consider the Transformer model trained with L3-iRMAC feature and Circle loss in the following experiments, denoted as CECL.

| Feature    | FIVR-200K | Loss | τ/γ | FIVR-200K |
|------------|-----------|------|-----|-----------|
|            | DSVR | CSVR | ISVR | DSVR | CSVR | ISVR |
| iMAC       | 0.547 | 0.526 | 0.447 | InfoNCE 0.07 | 0.493 | 0.473 | 0.394 |
| L3-iRMAC   | 0.570 | 0.553 | 0.473 | InfoNCE 1/256 | 0.566 | 0.548 | 0.468 |
| Circle     | 256   | 0.570 | 0.553 | 0.473 |

---

As in [34], we use two evaluation settings on CC_WEB_VIDEO, one measuring performance only on the query sets, and one on the entire dataset.
### 4.3 Ablation Study

In this section, we present the ablation study on different training mechanisms and similarity calculation methods. For the training mechanism, we compare the baseline (contrastive learning with memory bank [74]) with a triplet based approach with hard negative mining [36] and a modified MoCo [18]-like approach, where a large queue is maintained to store the negative samples and the weight of the model is updated in a moving averaged manner. For the triplet-based approach, the training process is extremely time-consuming (5 epochs, 5 hours on 32 Tesla-V100-SMX2-32GB GPUs), yet still show around 10% lower mAP compared with the baseline (40 epochs, 15 minutes on 4 Tesla-V100-SMX2-32GB GPUs), indicating that compared with learning from hard negatives, to utilize a large number of randomly sampled negative samples is not only more efficient, but also more effective. For the MoCo-like approach, we experimented it with different momentum (parameter $m$) ranging from 0.1 to 0.999, but none of them show better performance than the baseline approach as reported in Table 3a, we argue that the momentum mechanism is a compromise for larger memory, as the memory bank is big enough in our case, the momentum mechanism is not needed.

For the similarity measures, we evaluate both the aggregated video-level feature and the frame-level feature. For the video-level features, we evaluate with cosine similarity. For the frame-level features, we evaluate the similarity between frames with cosine similarity, and for the generated video-video similarity matrix, we calculate the similarity between videos following the setting of [34], i.e. chamfer similarity, symmetric chamfer similarity and chamfer similarity with similarity comparator (the weights are kept as provided by the authors). All four approaches are denoted as $\text{CECL}_c$ (cosine), $\text{CECL}_f$ (chamfer), $\text{CECL}_{sym}$ (symmetric-chamfer), $\text{CECL}_v$ (video comparator) for simplicity. Table 3b presents the results on FIVR-5K dataset. Interestingly, the frame-level similarity calculation approach outperforms the video-level approach by a large margin, indicating that frame-level comparison is important for fine-grained similarity calculation between videos. Besides, the comparator network does not show as good results as reported, we argue that this may be due to the bias between features. We did not re-train the comparator because our target is to learn a good video representation, and the similarity measure is expected to be as simple and computationally efficient as possible.

### 4.4 Comparison with the state-of-the-art

Near-duplicate Video Retrieval. We first compare the performance of CECL against state-of-the-art approaches on several versions of CC_WEB_VIDEO [72] following the setting in [34]. The benchmark approaches are Deep Metric Learning (DML) [36], the Circulant Temporal Encoding (CTE) [52], and Fine-grained Spatio-Temporal Video Similarity Learning (ViSil), we report the best results of the original paper. As listed in Table 5, for the aggregated video-level descriptor, we report the state-of-the-art result on all tasks, for the refined frame-level descriptor, we also report results comparable with ViSil$\_v$. To emphasize again, our target is to learn a good video representation, and the similarity calculation stage is expected to be as simple and computationally efficient as possible, therefore, it is fairer to compare our proposed approach with ViSil$\_v$, as they hold akin similarity calculation approach.

Fine-grained Incident Video Retrieval. For the FIVR task, we evaluate the performance of CECL against the state-of-the-art approaches on FIVR-200K [33] dataset. We report the best results reported in the original paper of Deep Metric Learning (DML) [36], Layer Bag-of-Words (LBoW) [35], Hashing Codes (HC) [60] and Fine-grained Spatio-Temporal Video Similarity Learning (ViSil) [34]. For models trained on VCDDB dataset, we report the result on FIVR-200K, and for models trained on FIVR-TRAIN, we report the result on FIVR-TEST. As shown in Table 6, still, the proposed feature aggregation approach show a clear performance advantage over state-of-the-art methods on video-level features (CECL$\_c$), and deliver competitive results when compared with frame-level features (CECL$\_f$) with low cost for similarity measure. Compared with ViSil$\_v$, we show a clear performance advantage even with a more compact frame-level feature and simpler frame-frame similarity measure, opening up new possibilities of incorporating contextual information of feature-level features. When compared with ViSil$\_f$, we show competitive results with much lower cost for similarity measure. Interestingly, our method slightly outperforms ViSil$\_f$ in ISVR task, indicating that our model might show an advantage in modeling semantic information. Besides, when trained on FIVR-TRAIN, all approaches see an improvement between 1% to 4%.

To make a fair comparison against the LBOV, we also report the results that FIVR-200K is used as both development dataset and evaluation dataset (only in this case, CECL$\_f$ is trained on the complete FIVR-200K dataset) in Table 7. Still, we present the best video-level feature, and the results in DSVR and CSVR task are further boosted by a large margin when the frame-level feature is used for similarity calculation. Besides, it is interesting that CECL$\_f$ show a clear performance advantage in ISVR task over CECL$\_f$, this may indicate that fine-grained frame-level comparison may be only effective for tasks that similar videos share visually similar scenes, and in terms of tasks that similar videos are only semantically similar, the video-level feature is more robust to visually similar distractor frames.

Event Video Retrieval. For EVR, we also compare CECL with the state-of-the-art approaches, i.e. Learning to Align and Match Videos (LAMV) [4] with Average Query Expansion (AQE) [13] and our old friend, Fine-grained Spatio-Temporal Video Similarity Learning (ViSil) [34] on EVVE [52]. We report the results of LAMV from

| Method | FIVR-5K |
|--------|---------|
|        | DSVR CSVR ISVR |
| baseline | 4096 0.609 0.617 0.578 |
| triplet | - 0.510 0.509 0.455 |
| m 0.1 | 65536 0.606 0.612 0.569 |
| m 0.9 | 65536 0.606 0.612 0.569 |
| m 0.9 | 65536 0.602 0.606 0.561 |
| m 0.999 | 65536 0.581 0.577 0.520 |

| Method | FIVR-5K |
|--------|---------|
|        | DSVR CSVR ISVR |
| CECL$\_c$ | 0.609 0.617 0.578 |
| CECL$\_f$ | 0.844 0.834 0.763 |
| CECL$\_sym$ | 0.763 0.766 0.711 |
| CECL$\_v$ | 0.726 0.735 0.701 |

| Method Bank FIVR-5K |
|---------------------|
| Size DSVR CSVR ISVR |
| baseline | 4096 |
| triplet | - |
| m 0.1 | 65536 |
| m 0.9 | 65536 |
| m 0.999 | 65536 |

| Method | FIVR-5K |
|--------|---------|
|        | DSVR CSVR ISVR |
| CECL$\_c$ | 0.609 0.617 0.578 |
| CECL$\_f$ | 0.844 0.834 0.763 |
| CECL$\_sym$ | 0.763 0.766 0.711 |
| CECL$\_v$ | 0.726 0.735 0.701 |
the original paper, and the re-evaluated ViSiL as the reported results are evaluated on around 80% of the original EVVE dataset. As shown in Table 4, CECLsym shown best over all mAP and some of the events, still competitive against ViSiL, that achieve best result on the majority of the events, but with much less computational cost. Surprisingly, our video-level feature version CECLc also report notable results, indicating that the temporal information and fine-grained spatial information are not necessary for event video retrieval task.

Table 8: Comparison on efficiency

| Method   | # Epochs | # GPU hours |
|----------|----------|-------------|
| Triplet  | 5        | 160         |
| Ours     | 40       | 1           |
| ViSiL [34] | 2211    |             |
| ViSiLc [34] | 2608    |             |
| CECLf    |          |             |

In Table 8, we demonstrate the efficiency of our method. For training, our method (contrastive learning with memory bank) is not only much efficient than the commonly used triplet-based approach, but also show significantly higher performance as reported in Table 3a. For evaluation, our method is about 22x faster comparing with ViSiL [34], while achieving competitive performance. All this shows that our method achieves a good trade-off between efficiency and performance, and holds great potential for application.

5 CONCLUSION

In this paper, we present CECL (Context Encoding for video retrieval with Contrastive Learning), a video representation learning network that aggregates the context information of frame-level descriptors. To train the feature aggregation models with pair-wise labels, we propose a supervised contrastive learning method, in which the models are trained to distinguish the positive sample with respect to the anchor sample from distractors contained in a shared memory bank with a contrastive loss. By conducting gradient analysis, the property of automatic hard negative mining is also discovered in the proposed method. Extensive experiments are conducted on multi video retrieval tasks, and the proposed method shows a clear performance advantage over state-of-the-art methods with video-level features and delivers competitive results with a much lower computational cost for similarity measure when compared with frame-level features.

1) Used in both DML [36] and ViSiL [34].
REFERENCES

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NeVLD. ieeec. 2017 conference on computer vision and pattern recognition. 25–34.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).

[3] Artem Babenko and Victor Lempitsky. 2015. Aggregating local deep features for image retrieval. In Proceedings of the IEEE international conference on computer vision. 1269–1277.

[4] Lorenzo Baraldi, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2018. Learning to align and match videos with kernelized temporal layers. In Proceedings of the 6th ACM international conference on Image and video retrieval. 2408–2415.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020).

[6] KyungHyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yosuf Bengio. 2014. On the properties of neutral machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014).

[7] Qiong Cao, Yiming Ying, and Peng Li. 2013. Similarity metric learning for face verification. Conference on Computer Vision and Pattern Recognition. 770–778.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1911.05722 (2019).

[9] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017).

[10] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Spatio-temporal contrastive learning via lifted structured feature embedding. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[11] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yosuf Bengio. 2014. On the properties of neutral machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014).

[12] Vincenzo De Ricco, Lorenzo Baraldi, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2018. Learning to align and match videos with kernelized temporal layers. In Proceedings of the IEEE international conference on computer vision. 2408–2415.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020).

[14] KyungHyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yosuf Bengio. 2014. On the properties of neutral machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014).

[15] Qiong Cao, Yiming Ying, and Peng Li. 2013. Similarity metric learning for face verification. Conference on Computer Vision and Pattern Recognition. 770–778.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1907.03748 (2018).

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In NeurIPS.

[18] Florent Perronnin and Christopher Dance. 2007. Fisher kernels on visual vocabularies for image categorization. In Proceedings of the 29th international conference on pattern recognition. IEEE, 3–20.

[19] Antonio Mignan, Mervin Gouedart, Jürgen Schmidhuber. 1997. Long short-term memory. Computer Vision and Pattern Recognition. 867–874.

[20] Hervé Jégou and Ondrej Chum. 2012. Nearest neighbour image search by compact quantisation. In Proceedings of the 6th ACM international conference on Image and video retrieval. 494–501.

[21] Weizhen Jin, XuanShan Nie, Chaorun Cui, Xiaoming Xi, Gongqing Yang, and Yulong Yin. 2019. Global-view hashing: harnessing global relations in near-duplicate video retrieval. World Wide Web 22, 2 (2019), 771–789.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1907.03748 (2018).

[23] Filip Radenović, Giorgos Tolias, and Ondřej Chum. 2016. CNN image retrieval networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition. 5588–5597.

[24] Qiong Cao, Yiming Ying, and Peng Li. 2013. Similarity metric learning for face verification. Conference on Computer Vision and Pattern Recognition. 770–778.

[25] Vincenzo De Ricco, Lorenzo Baraldi, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2018. Learning to align and match videos with kernelized temporal layers. In Proceedings of the IEEE international conference on computer vision. 2408–2415.

[26] Hervé Jégou and Ondřej Chum. 2012. Nearest neighbour image search by compact quantisation. In Proceedings of the 6th ACM international conference on Image and video retrieval. 494–501.
