CenterCLIP: Token Clustering for Efficient Text-Video Retrieval

Shuai Zhao
CCAI, Zhejiang University
Hangzhou, China
zhao19961111@gmail.com

Linchao Zhu
ReLER Lab, AAII, University of Technology Sydney
Sydney, Australia
zhulinchao7@gmail.com

Xiaohan Wang
CCAI, Zhejiang University
Hangzhou, China
wxh19961111@gmail.com

Yi Yang
CCAI, Zhejiang University
Hangzhou, China
yangyics@zju.edu.cn

ABSTRACT
Recently, large-scale pre-training methods like CLIP have made great progress in multi-modal research such as text-video retrieval. In CLIP, transformers are vital for modeling complex multi-modal relations. However, in the vision transformer of CLIP, the essential visual tokenization process, which produces discrete visual token sequences, generates many homogeneous tokens due to the redundancy nature of consecutive and similar frames in videos. This significantly increases computation costs and hinders the deployment of video retrieval models in web applications. In this paper, to reduce the number of redundant video tokens, we design a multi-segment token clustering algorithm to find the most representative tokens and drop the non-essential ones. As the frame redundancy occurs mostly in consecutive frames, we divide videos into multiple segments and conduct segment-level clustering. Center tokens from each segment are later concatenated into a new sequence, while their original spatial-temporal relations are well maintained. We instantiate two clustering algorithms to efficiently find deterministic medoids and iteratively partition groups in high dimensional space. Through this token clustering and center selection procedure, we successfully reduce computation costs by removing redundant visual tokens. This method further enhances segment-level semantic alignment between video and text representations, enforcing the spatio-temporal interactions of tokens from within-segment frames. Our method, coined as CenterCLIP, surpasses existing state-of-the-art by a large margin on typical text-video benchmarks, while reducing the training memory cost by 35% and accelerating the inference speed by 14% at the best case. The code is available at https://github.com/mzhaoshuai/CenterCLIP.

CCS CONCEPTS
• Information systems → Novelty in information retrieval.

KEYWORDS
Text-video retrieval; CLIP; transformer; token clustering

1 INTRODUCTION
Text-video retrieval is less studied than the commonly known text-image retrieval task as the intricate context of the video, especially when the length of the video is very long or the temporal variation of the video is large. With the explosive growth of video content on mobile phones and Internet during the past decade, text-video retrieval becomes increasingly popular. People also desire a better text-video retrieval system as searching for videos of interest already becomes a part of daily lives of most people.
Recently, with the success of large-scale contrastive language-image pre-training methods like CLIP [39], text-video retrieval also has made great progress. To be specific, CLIP4clip [32] transfers the knowledge of CLIP to text-video retrieval tasks, surpassing the previous state-of-the-art methods by a large margin (e.g., more than 30% improvement of the recall metric on ActivityNet [12]). This demonstrates the power of billion-scale image-text pairs pre-training via contrastive learning. In CLIP, a vision transformer [11, 51] is adopted for visual representation learning. Typically, in vision transformer, visual tokenization, i.e., linear projection of non-overlapped image patches to an embedding space, is a necessary component to produce discrete visual token sequences. Then token sequences can be processed by the multi-head self-attention (MHSA) in transformer blocks as the same manner of dealing with text sequences in the original transformer [51].

When the input of the vision transformer becomes videos, the visual tokenization procedure produces many homogeneous tokens due to the redundancy nature in continuously changing frames. In Figure 1, we extract the token embedding of CLIP from different frames in the same video and visualize them by t-SNE [50]. From the visualization, we can see those token embeddings from different frames form many tight clusters. Image patches with similar texture features correspond to immediate data points within a certain cluster. It is also clear that the number of clusters and the average number of tokens in clusters are not small, i.e., there are many similar token embedding in high-dimensional space. As a result, repeated computation of these homogeneous tokens in CLIP inevitably introduces a lot of unnecessary computation costs and hinders the training and deployment of video retrieval models in web applications. To resolve the above problem, in this work, we propose to distinguish the most representative tokens, i.e., the center token of each cluster in Figure 1, and only use these typical tokens for visual representation learning as these tokens contribute most to the discriminative feature representation learning.

We introduce a multi-segment token clustering algorithm to find the most representative tokens to reduce computation costs, and achieve segment-level semantic alignment of video and text representation. An input video is divided into multiple temporal segments. Each segment contains the same number of consecutive frames. Given the token embeddings of these frames, a clustering algorithm is performed on each segment independently. After clustering, only center tokens of clusters are reserved and non-center tokens are dropped to avoid duplicated computation of similar tokens. This significantly reduces computation costs. Center tokens from the same temporal segment are then concatenated into a new visual sequence and arranged according to their original spatial-temporal positions, i.e., tokens whose image patches occur earlier in the video would appear at the earlier position of the new visual sequence. Then the new visual sequence is processed by the standard transformer blocks. This enables the model to learn segment-level video representation via attention among tokens from within-segment frames. These segment-level video representations are aligned with the text through contrastive learning.

In this work, we introduce two instances of clustering algorithm in multi-segment token clustering. One is k-medoids equipped with a deterministic centroids initialization method, i.e., KYZ initialization [23, 48], to ensure the clustering results are consistent through multiple runs. A good initialization also helps the clustering algorithm converge fast. The other is spectral clustering which suits high-dimensional data points clustering. With our multi-segment token clustering algorithm, CenterCLIP achieves state-of-the-art performance on four common benchmarks: MSR-VTT [60], MSVD [7], LSCMC [42], and ActivityNet [12]. We achieve significant improvement of retrieval metrics on all these four datasets compared to the baseline. At the same time, we achieve a decent reduction in memory cost and speed up the inference process. Specifically, on ActivityNet, we achieve a 35% reduction in memory cost and 14% speedup of inference speed compared to the baseline.

2 RELATED WORKS

Contrastive Vision-Language Pre-Training. Since the success of derivative works of Contrastive Language-Image Pre-Training (CLIP) [39] in different areas [4, 32, 37, 47, 59], visual representation learning under text supervision attracts widespread attention. Huge models pre-trained on billion-scale image-text pairs from web like WenLin [18], Google’s ALIGN [21], and Microsoft’s Florence [65] emerged. In the language-video understanding area, there are similar works like Frozen in Time [2] and HowTo100M [35]. However, the scale of language-video pre-training is much smaller than language-image pre-training as the former is much more expensive. Following CLIP4clip [32], we transfer the knowledge of CLIP to the text-video retrieval task in this work.

Text-video Retrieval. Text-video retrieval is more complex than commonly studied text-image retrieval as the additional temporal dimension introduces complex context information. Previously, standard language-video learning methods tend to design dedicated fusion manners for cross-model learning from offline extracted video and text features [20, 24, 26, 58, 62, 64]. Recently, the paradigm of end-to-end large-scale pre-training plus task-specific finetune becomes more and more popular for language-video understanding, e.g., HowTo100M [35], MIL-NCE [34], ActBERT [69], VideoBERT [49], MMT [13], and HERO [28]. These methods achieve promising results on many language-video tasks and demonstrate the effectiveness of pre-training. Our work is also in this line, the difference is that we inherit the knowledge from CLIP [39], which is pre-trained on image-text pairs rather than video-text pairs.

Efficient Transformer. Recently, transformer becomes the unified model for many vision and text tasks [11, 19, 30, 40, 51]. However, there are many time-consuming operations in transformers such as self-attention and softmax operations. Some works try to reduce the complexity of self-attention for very long sequences or remove the softmax operation, e.g., Performer [9], Linear Transformer [22], Linformer [55], Reformer [25], Sparse Transformer [8], Routing Transformer [44], Longformer [3], and Galerkin Transformer [6]. Very recently, in computer vision, people also notice that not all tokens matter for the final performance of the model. To reduce the computation cost, researchers try to learn a few most representative visual tokens [45, 57], learn to rank all tokens and select the most important ones [53], and learn to mask the unimportant tokens [41, 61]. Compared to these mentioned methods, we are parameter-free and introduce segment level semantic alignment of text and video representation for text-video retrieval.
3 METHODS

3.1 Preliminary

Given a video set \( V \) and text set \( T \), the goal of text-video retrieval is to learn a score function \( f \), which gives a high similarity score \( f(v_i, t_j) \) if a video \( v_i \in V \) and a text \( t_j \in T \) are highly relevant and a low similarity score for an irrelevant video-text pair. Then we can rank videos according to the query text (text to video retrieval) or rank texts according to the query video (video to text retrieval).

Following the typical multi-modal retrieval frameworks [27, 39], our text-video retrieval model is composed of a text encoder \( g \) and a video encoder \( h \). Given a text \( t_j \) and a video \( v_i \), \( g(t_j) \) and \( h(v_i) \) produce the normalized high-dimensional feature of the input, where \( \ell_2 \) normalization is often considered in final feature encoding. Then the similarity score of this text-video pair \((v_i, t_j)\) is

\[
 f(v_i, t_j) = h(v_i)^T g(t_j). \tag{1}
\]

In training, a video-text pair \((v_i, t_j)\) is treated as the positive if \( v_i \) and \( t_j \) are corresponded. All other instances of video or text in the mini-batch are treated as the negative. The text and video encoders are optimized in an end-to-end manner via normalized softmax loss [66]. The overall loss \( L \) is the average of video-to-text classification loss \( (L_{vt}) \) and text-to-video classification loss \( (L_{tv}) \):

\[
 L_{vt} = -\frac{1}{N} \sum_i^N \log \frac{\exp(g(t_j)^T h(v_i)/\tau)}{\sum_{j=1}^N \exp(g(t_j)^T h(v_i)/\tau)}, \tag{3}
\]

\[
 L = \frac{1}{2} (L_{vt} + L_{tv}), \tag{4}
\]

where \( N \) is the mini-batch size and \( \tau \) is the temperature to scale the logits. It is worth noting that \( \tau \) is crucial because both \( h(v_i) \) and \( g(t_j) \) are normalized. We set it as a trainable parameter following the CLIP model. During training, our model is initialized from the pre-trained weight of CLIP. We describe the details of the text encoder and video encoder below.

3.1.1 Text encoder. We instantiate the text encoder using the text model of CLIP. It is a transformer [51] with the architecture modifications described in BERT [40], i.e., only encoder and no decoder. A transformer model typically consists of repeated blocks (layers) of multi-head self-attention (MHSA) and feed-forward networks (FFN). We use a transformer with 12 layers and 512 width with 8 attention heads, where the width is the dimension of the query, key, and value feature. The text tokenizer is a lower-cased byte pair encoding (BPE) [46] with a 49 152 vocab size. The text sequence is padded with [SOS] and [EOS] tokens. [SOS] and [EOS] is padded at the beginning and end of the text sequence, respectively. The final text feature representation is the activation from the last layer of the transformer that corresponds to the [EOS] token. This text representation is later normalized by layer normalization and linearly projected into the joint video-text embedding space.
3.1.2 **Video encoder.** Our video encoder is a vision transformer (ViT), which first successfully applied transformers in vision tasks. The architecture of ViT is the same as the transformer in natural language processing, except ViT introduces an additional visual tokenization process to convert images into discrete sequences. When feeding images or videos into a ViT, we first convert the non-overlapped image patches into visual tokens, where a [CLASS] token is prepended to the beginning of sequences as BERT [40]. Then the output of [CLASS] token at the final layer is extracted as the visual representation. In this work, we adopt a 2D linear projection to project image patches of different frames into an embedding space independently following the practice of CLIP4clip [32]. For convenience, we name this linear transformation process as visual tokenization. Generally, we use a ViT-B/32 model [11] with 12 layers and 512 width with 8 attention heads. ViT-B/32 means the non-overlapped input image patch size is $32 \times 32$.

When applying the visual tokenization process to videos, it inevitably produces many redundant tokens as shown in Figure 1. Generally, an input video $v_i$ consists of many temporal related frames: $v_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$, where $|v_i|$ is the number of frames in $v_i$. After visual tokenization, if each frame produces $L$ tokens, the number of visual tokens is $L|v_i|$ (do not consider [CLASS] token). It shows that the number of visual tokens is linear to the number of tokens per frame ($L$) and the video length. Given an input frame with a size of $224 \times 224$, $L = 49$ for the ViT-B/32 model and $L = 196$ for the ViT-B/16 model. With a larger $L$, the number of visual tokens becomes much larger. When performing text-video retrieval on long videos, the total number of tokens for a video is large. For example, videos in the ActivityNet [12] dataset usually have a few minutes duration. In this case, $L|v_i|$ will be easily larger than 1,000.

The redundant tokens considerably increase computation costs. To make the training and inference of text-video retrieval models more efficient, we propose to use clustering algorithms to find the most representative token embeddings. This process significantly reduces the number of tokens while maintaining the most valuable information of original tokens. After clustering, we only reserve the center tokens and remove other non-center tokens. The reserved tokens contain most of the information about the video and it is sufficient for text-video retrieval. We describe our multi-segment token clustering in the next section.

3.2 **Multi-segment Token Clustering**

The overall framework of our video encoder can be found in Figure 2. We perform a multi-segment clustering strategy on visual tokens from a certain temporal segment. This is based on the assumption that neighbor frames are more likely to be the same; then tokens of these similar neighbor frames are more possible to be redundant. Our multi-segment token clustering method empowers the model to achieve segment-level semantic alignment of text and video representations. Previously, CLIP and CLIP4clip adopt the average of frame features or the late fusion of frame features as the video representation. However, the former loses the temporal information, and the latter is a post-processing step and loses the detail temporal variations at the early stage of the transformer. By clustering across multiple frames within a segment at an early or middle stage, image patches from different temporal positions can interact with each other via the self-attention mechanism.

Specifically, a video sequence $\{v_{i1}, v_{i2}, \ldots, v_{in}\}$ is divided into $S$ segments $\{s_{1i}, s_{2i}, \ldots, s_{si}\}$. Each segment contains $\frac{L|v_i|}{S}$ frames and $L|v_i|$ tokens. Then we perform token clustering on these $\frac{L|v_i|}{S}$ tokens segment-wise, namely, clustering for each segment independently. Then the centers of all clusters from one segment, i.e., center tokens, are selected and other non-center tokens are simply dropped. These center tokens are concatenated and arranged according to their original relative spatial-temporal position. Center tokens from the upper-left position and early frames are at the beginning of the new token sequence. Center tokens from the bottom-right position and late frames are at the rear-end of the new token sequence.

Multi-segment token clustering algorithm makes our vision model achieve segment-level temporal modeling and be able to capture the detailed temporal variation of video frames. This allows our methods to achieve segment-level alignment of the text $t_i$ and the video $v_i$ consisted of segments $\{s_{1i}, s_{2i}, \ldots, s_{si}\}$:

$$f(v_i, t_i) = \frac{1}{S} \sum_{j=1}^{S} h(s_{ji})^T g(t_i).$$

(5)

The multi-segment token clustering method has at least two advantages: (1) reducing computation costs by cutting down the number of tokens; (2) achieving segment-level semantic alignment of text and video representations via attention among tokens from different frames within the same temporal segment. As shown in Figure 2, assuming we perform token clustering right after the $B$-th transformer block and the number of clusters is $K$ (ignore [CLASS] token after pooling), this means the length of the input sequence length of the following $(12 - B)$ transformer blocks become $K$. Generally, $\frac{L|v_i|}{S} \gg K$, obviously, computational costs are largely reduced. It is worth noting that the clustering module can be inserted at any place of ViT and the clustering procedure can be performed for any times. Clustering at an early stage reduces more computation costs. Next, we discuss two clustering methods used in the multi-segment token clustering algorithm.

3.2.1 **k-medoids++.** In this section, we introduce the first instance of the token clustering method: k-medoids++, a variety of commonly known k-means algorithm. Given a set of tokens $\{x_1, \ldots, x_m\} \in \mathbb{R}^d$, where $d$ is the transformer width and $m = \frac{L|v_i|}{S}$ is the number of tokens in a temporal segment, the goal of k-means is to partition the $m$ observations into $K$ ($\leq m$) sets, i.e., $C = \{C_1, C_2, \ldots, C_K\}$, so as to minimize the within-cluster sum of squares: $\arg\min_C \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2_2$, where $\mu_i$ is the mean of points in $C_i$, i.e., centroid. Normal k-means contains 4 steps:

1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^d$ randomly;
2. For every $i$, set $p_i := \arg\min_j ||x_i - \mu_j||^2_2$;
3. For every $j$, set $\mu_j := \frac{\sum_{i=1}^{m} 1\{p_i = j\} x_i}{\sum_{i=1}^{m} 1\{p_i = j\}}$; $1\{\cdot\}$ equals to 1 if and only if the inner condition is true;
4. Repeat step 2 and 3 until convergence.

One disadvantage of the normal k-means is that clustering results are sensitive to the centroids initialization. Bad initialization may lead to the collapse of the clustering. Random initialization is
also not suitable for retrieval, as we would obtain inconsistent retrieval results when querying multiple times. Therefore, we need a deterministic centroids initialization method. In this work, we adopt the KIT initialization method [23, 48]. The algorithm is shown in Algorithm 1. It first chooses the point with the maximum $\ell_2$-norm as the centroid, then chooses the point with the maximum distance to the existing centroids as the next centroids. The algorithm is simple but effective [23, 48]. It makes k-means deterministic and accelerates its convergence speed.

In our token clustering process, we use medoids rather than centroids. Namely, we choose the nearest point to centroids as the seed point in steps 2K3 in the normal k-means methods. This is because the semantic of mean pooling representation of tokens within a cluster may shift away from the exact token embeddings. Combining k-medoids and KIT initialization, we get k-medoids++ — the name philosophy follows k-mean++ [1]. The complexity of k-medoids++ is $O(mKdI)$, where $I$ is the iteration upper bound and $O(d)$ for computing the distance between two points in $\mathbb{R}^d$.

3.2.2 Spectral clustering. K-means is suitable for spherical data clusters. However, our data points are in a high dimension space $\mathbb{R}^d$ ($d=512$ in most cases), and the data distribution is unknown. The shape of data clusters may not be spherical. To resolve this method that aims to maximize the weights of connections within groups and minimize the weights of connections between groups. It first needs to construct the graph $G = (X, E)$, where $X$ is the vertex set and each vertex represents a data point $x$. $E$ is the edge set and each edge denotes the (weighted) connection between two vertices. In this work, we use the normalized spectral clustering algorithm described in [56]. The algorithm contains 5 steps:

1. Construct similarity graph. Let $W$ be its weighted adjacency matrix, $D$ be the degree matrix;
2. Compute normalized Laplacian $L_{sym} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$;
3. Compute the first $K$ eigenvectors $\mu_1, \ldots, \mu_K$ of $L_{sym}$ which correspond to the first $K$ least eigenvalues;
4. Let $U = [\mu_1, \ldots, \mu_K]$; Normalize each row of $U$ to have norm of 1, generally, $\ell_2$ norm is used;
5. Consider each row of $U$ as a new data point, apply k-means to these data points.

The above algorithm first performs dimension reduction and then data clustering. In this work, we use SVD to solve the eigenvectors of $L_{sym}$ as $L_{sym} = L_{sym}$. A sign correct algorithm is further introduced to resolve the sign ambiguity in SVD as the direction of points after dimension reduction also matters for some distance metrics, e.g., $\ell_2$ distance. The main idea of this sign correct algorithm is to make the direction of singular vectors aligns with the majority direction of data points [5], i.e., the sign of the sum of the inner product of singular vectors and data points should be positive. Here we describe the special case of this algorithm for symmetric matrix in Algorithm 2. The complexity of spectral clustering is $O(mK^2I + m^3)$, where $O(m^3)$ for step 1-4 and $O(mK^2I)$ for step 5. For more details about spectral clustering refer to the tutorial [52].

4 EXPERIMENTS
4.1 Experimental Details

Dataset. We validate our model on four datasets: MSR-VTT [60], MSVD [7], LSMDC [42], and ActivityNet [12]. To save computational costs, the shorter side of videos are resized to 224 and the frame per second (fps) is set to 3. (a) MSR-VTT contains 10 000 videos with a length ranges from 10 ~ 32 seconds and 200 000 captions. We use two types of data splits, training-7K and training-9K, to compare with baselines. The training-7K follows the data splits from HowTo100M [35] and the training-9K follows the data splits from [13]. The test data in both splits is 'test 1k-A', which contains 1 000 video-text pairs following JSFusion [63]. If we do not specify, we use training-9K as the default. (b) MSVD contains 1 970 videos with a duration ranges from 1 ~ 62 seconds. Train, validation, and test splits contain 1 200, 100, and 670 videos, respectively. Each video has approximately 40 associated sentences in English. (c) LSMDC is comprised of 118 081 videos that ranges from 2~30 seconds. The videos were extracted from 202 movies. The validation set contains 7 408 videos. The 1 000 videos in the test set are from movies independent from the training and validation splits. (d) ActivityNet [12, 17] consists of 20 000 YouTube videos, and some of them are minutes long. We follow [13, 67] to concatenate all the
The document contains tables and text discussing performance metrics for different models on ActivityNet and MSVD datasets. The tables compare the speed and memory usage of various models, including MeM, CenterCLIP, and FSE. The text provides insights into learning strategies, optimization techniques, and the evaluation metrics used. The tables include columns for Speed (GB and ms), Recall at different ranks (R@1, R@5, R@10), and Median Rank (MdR). The metrics are compared across different datasets and configurations, highlighting performance gains with ViT-B/32 and ViT-B/16 architectures. The text concludes by discussing frame sampling strategies and the importance of maintaining a balance between model complexity and performance. The document provides a comprehensive overview of the architectural and experimental aspects of the models discussed.
### Table 3: Results on MSR-VTT. MeM. is the average GPU memory cost when training on 2 and 8 Tesla V100 GPUs for ViT-B/32 and ViT-B/16, respectively. Speed is the inference time per video during evaluation on a Tesla V100 GPU.

| Method | MeM (GB) | Speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| CLIP4clip (meanP) [32] | 20.8 | 24.4 | 42.1 | 71.9 | 81.4 | 2 | 15.7 | - | - | - | - | - |
| CLIP4clip (seqTransf) | - | - | 42.0 | 68.6 | 78.7 | 2 | 16.2 | - | - | - | - | - |

**Baseline (CLIP4clip (meanP), ViT-B/32)**

| Method | MeM (GB) | Speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| CenterCLIP (k-medoids++, B₂₄ = 4, 49) | 15.0 | 22.9 | 43.7 | 71.3 | 80.8 | 2 | 16.9 | 41.8 | 68.9 | 77.9 | 2 | 13.3 |
| CenterCLIP (k-medoids++, B₂₄ = 3, 49) | 14.2 | 22.9 | 43.5 | 68.5 | 79.7 | 2 | 17.7 | 40.9 | 68.4 | 78.3 | 2 | 13.4 |
| CenterCLIP (spectral, B₂₄ = 4, 49) | 14.9 | 40.8 | 43.4 | 70.5 | 79.8 | 2 | 15.7 | 42.1 | 70.5 | 80.6 | 2 | 11.7 |
| CenterCLIP (spectral, B₂₄ = 3, 49) | 14.2 | 43.6 | 43.7 | 71.3 | 80.2 | 2 | 16.2 | 43.2 | 71.0 | 80.4 | 2 | 12.3 |

**CenterCLIP (meanP), ViT-B/16**

| Method | MeM (GB) | Speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| ActBERT [69] | - | - | 8.6 | 23.4 | 31.1 | 36 | - | - | - | - | - |
| JSFusion [63] | - | - | 10.2 | 31.2 | 43.2 | 13 | - | - | - | - | - |
| HowTo100M [35] | - | - | 14.9 | 40.2 | 52.8 | 9 | - | - | - | - | - |
| CE [29] | - | - | 20.9 | 48.8 | 62.4 | 6 | - | 20.6 | 50.3 | 64.0 | 5.3 | - |
| MMT [13] | - | - | 26.6 | 57.1 | 69.6 | 4 | 24.0 | 27.0 | 57.5 | 69.7 | 3.7 | 21.3 |
| T2VLAD [56] | - | - | 29.5 | 59.0 | 70.1 | 4 | - | 31.8 | 60.0 | 71.1 | 3 | - |
| AVNet [43] | - | - | 27.1 | 55.6 | 66.6 | 4 | - | 28.5 | 58.6 | 71.6 | 3 | - |
| T2-CE [10] | - | - | 29.6 | 61.6 | 74.2 | 3 | - | 32.1 | 62.7 | 75.0 | 3 | 61.6 |
| CLIP zero-shot | - | - | 31.2 | 53.7 | 64.2 | 4 | - | 27.2 | 51.7 | 62.6 | 5 | - |
| CLIP4clip (meanP) [32] | 20.8 | 24.4 | 43.1 | 70.4 | 80.8 | 2 | 16.2 | 43.1 | 70.5 | 81.2 | 2 | 12.4 |
| CLIP4clip (seqTransf) | - | - | 44.5 | 71.4 | 81.6 | 2 | 15.3 | 42.7 | 70.9 | 80.6 | 2 | 11.6 |

### Table 4: Results on LSMDC. MeM. is the average GPU memory cost when training on 2 and 8 Tesla V100 GPUs for ViT-B/32 and ViT-B/16, respectively. Speed is the inference time per video during evaluation on a Tesla V100 GPU.

| Method | MeM (GB) | speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| JSFusion [63] | - | - | 9.1 | 21.2 | 34.1 | 36.0 | - | 12.3 | 28.6 | 38.9 | 20.0 | - |
| CE [29] | - | - | 11.2 | 26.9 | 34.8 | 25.3 | 96.8 | - | - | - | - | - |
| MMT [13] | - | - | 12.9 | 29.9 | 40.1 | 19.3 | 75.0 | 12.3 | 28.6 | 38.9 | 20.0 | 76.0 |
| TT-CE- [10] | - | - | 15.0 | 30.8 | 39.8 | 20.0 | - | - | - | - | - | - |
| CLIP zero-shot | - | - | 17.2 | 36.5 | 46.3 | 13.7 | - | 17.5 | 36.0 | 45.0 | 14.3 | - |
| CLIP4clip (meanP) [32] | 20.8 | 24.4 | 20.7 | 38.9 | 47.2 | 13.0 | 65.3 | 20.6 | 39.4 | 47.5 | 13.0 | 56.7 |
| CLIP4clip (seqTransf) | - | - | 22.6 | 41.0 | 49.1 | 11.0 | 61.0 | 20.8 | 39.0 | 48.6 | 12.0 | 54.2 |

**Baseline (CLIP4clip (meanP), ViT-B/32)**

| Method | MeM (GB) | speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| CenterCLIP (k-medoids++, B₂₄ = 4, 49) | 16.4 | 23.9 | 21.9 | 41.1 | 50.7 | 10.0 | 55.6 | 21.1 | 41.2 | 50.2 | 10.0 | 48.7 |
| CenterCLIP (k-medoids++, B₂₄ = 3, 49) | 15.0 | 22.9 | 21.7 | 39.8 | 49.8 | 11.0 | 54.8 | 21.4 | 40.3 | 50.8 | 10.0 | 48.4 |
| CenterCLIP (spectral, B₂₄ = 4, 49) | 16.4 | 40.8 | 21.6 | 40.9 | 49.3 | 11.0 | 57.2 | 20.6 | 39.5 | 48.8 | 12.0 | 51.4 |
| CenterCLIP (spectral, B₂₄ = 3, 49) | 15.0 | 43.6 | 21.4 | 39.7 | 49.4 | 11.0 | 55.9 | 19.5 | 39.9 | 48.0 | 12.0 | 50.1 |

**Baseline (CLIP4clip (meanP), ViT-B/16)**

| Method | MeM (GB) | speed (ms) | Video → Text | Text → Video |
|--------|-----------|------------|---------------|---------------|
|        | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ | R@1↑ | R@5↑ | R@10↑ | MdR↓ | MnR↓ |
| CenterCLIP (k-medoids++, B₂₄ = 4, 49) | 25.7 | 59.6 | 24.1 | 45.0 | 55.1 | 8 | 51.1 | 22.5 | 42.9 | 53.5 | 9 | 45.1 |
| CenterCLIP (k-medoids++, B₂₄ = 3, 49) | 17.6 | 86.5 | 24.2 | 46.2 | 55.9 | 8 | 47.3 | 24.5 | 46.4 | 55.8 | 7 | 41.3 |
number of clusters/centers is constant $K$. Generally, we construct KNN graph with Gaussian similarity function between two points when applying spectral clustering: $\exp(-|x_i - x_j|^2/(2\sigma^2))$. The neighbours of one vertex is $5K$ (the number of frames in a segment) for ViT-B/32, and plus an additional 5 for ViT-B/16. The variance of the Gaussian function $\sigma$ is simply set to 2.0. No normalization is applied for token embeddings before performing clustering. Baselines in the experiments use the same setting as CenterCLIP.

### 4.2 Results on Common Benchmarks

As shown in Table 1, Table 2, Table 3, and Table 4. We achieve SOTA performance on all four datasets. Moreover, we also achieve decent memory usage reduction in all cases and obvious speedup of evaluation in some cases. Specifically, for CenterCLIP (ViT-B/32), we achieve a 32% reduction in memory cost and accelerate the model by 6% of the original speed for MSR-VTT, MSVD, and LSMDC in the best situation. For ActivityNet, the reduction in memory cost is 35% and the speedup of evaluation speed is 14% in the best case. These numbers verify the efficiency of our method. For CenterCLIP (ViT-B/16), as the patch size decreases, the number of tokens increases ($4 \times$ as the number of tokens of ViT-B/32). In this work, the clustering complexity is at least linearly related to the number of data points. Therefore, CenterCLIP does not gain speedup for ViT-B/16. However, CenterCLIP also achieves a 32% reduction in memory cost. In future work, we will introduce faster clustering strategies to speed up the whole model.

Compared to the baseline, CenterCLIP achieves significant improvement on recall. When using ViT-B/32, for MSVD, the maximal gain of text→video R@1 is 1.7%; for MSR-VTT (training-9K), the number is 1.2%; for LSMDC, it is 1.8%; for ActivityNet, it achieves 2.1% improvement of text→video R@1. When using ViT-B/16, for MSVD, the numbers are 1.0%, 2.8%, and 0.1% for text→video R@1, 5.7%, 5.2%, and 2.0% for video→text R@1. CenterCLIP gains more improvement of video→text retrieval performance in this case. All these results demonstrate the effectiveness of our clustering strategy. It aligns segment semantics of text and video.

It is worth noting that spectral clustering and k-medoids++ achieve similar performance in most cases. This is somehow counterintuitive as spectral clustering should be more suitable for clustering high-dimensional points. This is possible because the data shape of clusters of token embeddings in high dimensional space is nearly spherical. Spectral clustering does achieve better performance in terms of some metrics, e.g., better R@5 and R@10 on MSR-VTT and ActivityNet, and produces the best video→text results on MSVD.

### 4.3 Diagnostic Experiments

In this section, we will analyze CenterCLIP thoroughly. All diagnostic experiments are taken with CenterCLIP (ViT-B/32).

#### 4.3.1 More baselines

We provide four more strong baselines: 1) pooling of nearby tokens in a temporal segment, after pooling, we get $K$ average tokens for one segment; 2) sparse sampling of tokens in a temporal segment, namely, randomly sample $K$ tokens from a temporal segment during training and uniformly sample $K$ tokens during validation; 3) temporal shift described in TSM [54], here we apply temporal shift to the tokens except [CLASS] embedding; 4) token shift described in [68], the method only shift the [CLASS] embedding. The shift is performed twice in each transformer block, right before MHSA and FFN. Results are shown in Table 5. Shifting all image patches does not work here. Sparse sampling produces a little better results than baseline on MSR-VTT and LSMDC. However, CenterCLIP is much better than sparse sampling, this demonstrates the necessity of selecting representative tokens. Tokenshift achieves pretty good performance on short videos, nevertheless, it does not reduce any computational costs.

#### 4.3.2 The place of performing token clustering

The influence of places of token clustering (k-medoids++) is shown in Figure 3. The smaller the $B$, the lower the memory cost. The performance of the whole model will also decrease along with the decreasing or increasing of $B$. A good trade-off between memory cost and performance achieves at $B = 6$, this is also our default setting. We can also take multiple times of clustering. For instance, firstly perform clustering with $(S = 6, B = 4)$ and then with $(S = 3, B = 4)$. 

![Figure 3: Influence of places of token clustering.](image-url)

**Table 5: Comparison with strong token selection baselines.**

| Method                  | Mem. | $T \rightarrow V$ | R@5 | R@10 | R@30 | MdR | MnR |
|-------------------------|------|-------------------|------|------|------|-----|-----|
| MSR-VTT (train on training-7K) |
| pooling ($B_0 = 6, 49$) | 16.39 | 14.95 V | 41.9 | 66.6 | 76.7 | 2  | 18.7 |
| pooling ($B_0 = 4, 49$) | 14.95 | 16.39 V | 40.6 | 66.9 | 76.5 | 2  | 17.5 |
| sparse sampling ($B_0 = 6, 49$) | 16.39 | 14.95 V | 42.6 | 68.4 | 78.4 | 2  | 17.6 |
| sparse sampling ($B_0 = 4, 49$) | 14.95 | 16.39 V | 41.6 | 68.3 | 77.5 | 2  | 12.8 |
| token shift [68] | 20.77 | 20.77 V | 42.5 | 68.5 | 78.6 | 2  | 16.4 |
| temporal shift [54] | 20.77 | 20.77 V | 43.3 | 70.1 | 80.8 | 2  | 12.2 |
| CenterCLIP ($B_0 = 4, 49$) | 14.95 | 16.39 V | 43.7 | 71.3 | 80.8 | 2  | 16.9 |
| CenterCLIP ($B_0 = 6, 49$) | 16.39 | 14.95 V | 41.8 | 70.2 | 79.6 | 2  | 11.3 |

**Table 5:** Comparison with strong token selection baselines.
8). The results are shown in Table 6. Such progressive clustering strategy achieves pretty good R@5, R@10, MdR, and memory cost reduction. However, performing multiple times will increase the time complexity and this is not suitable for large amounts of tokens. Thus we generally perform clustering once in this work.

4.3.3 The number of cluster $K$ and segment $S$. We perform experiments on LSMDC and ActivityNet. The results including R@1, memory cost, and inference time are shown in Figure 4. Along with the increase of $K$, the performance increases, and computation costs also increase. At the same time, a small segment number $S$ does not always achieve better performance, e.g., $S = 1$ on LSMDC and $S = 6$ on ActivityNet. A small segment number means more tokens are dropped. This will cause the loss of more information. When $S = 1$ on LSMDC and $S = 6$ on ActivityNet, the number of tokens in a segment is large, i.e., $12 \times 49$ and $10 \times 49$, this leads to more computational costs of clustering as shown in Figure 4c and Figure 4d. Thus a moderate segment number $S$ is usually adopted.

4.3.4 The number of input frames $N_{in}$. We change the number of input video frames and take experiments with CenterCLIP ($B_6$ – 15, 49) on ActivityNet. The results are shown in Table 7. The large number of input frames $N_{in}$, the more computation costs, and a small number of frames will lead to worse performance. Similar ablations about the input of frames on short video datasets like MSR-VTT can be found in CLIP4clip [32]. When the number of segments $S$ is fixed, a large number of input frames $N_{in}$ will also increase computation costs of the clustering process as the number of tokens in one temporal segment increases.

4.3.5 Normalization of token embeddings. People may be curious about the influence of embedding normalization when performing token clustering. We showed results with and without $l_2$ normalization on LSMDC.

4.3.6 Learning rate and training epochs. The original CLIP4clip uses a learning rate of 1e-7. When setting $l_r = 1e-7$ on MSR-VTT (training-7K), we get 39.7 T$\rightarrow$V R@1 with mixed precision [33] and 41.7 T$\rightarrow$V R@1 without mixed precision on the split ‘test 1k-A’. The corresponding result of CLIP4clip baseline is 42.1 T$\rightarrow$V R@1. When increasing the learning rate to 5e-6, we get 42.4 T$\rightarrow$V R@1 with mixed precision on ‘test 1k-A’. As the mixed precision training saves a lot of GPU memory and accelerates the training procedure,
The four-legged creature, half-horse, half-eagle, rises above the trees.

Figure 5: Visualization of centers after token clustering with different number of frames in a temporal segment.

we are stuck with mixed precision and use $lr = 5e-6$ for short video datasets. For ActivityNet, we found a large learning rate with more training epochs brings a better result. This is shown in Table 9. It is possibly because of the large number of different video frames in long videos in ActivityNet and the sparse sampling strategy we used during training. The model needs more training epochs to learn good representations of videos.

4.3.7 Visualization of image patches of center tokens after clustering.

We further display visualization results of center tokens after multi-segment clustering with different numbers of video frames within a temporal segment. The results are shown in Figure 5. It is clear that the clustering algorithm reserves the most representative tokens, for example, in the second and third row of Figure 5, tokens of the foreground animal are selected and only part of the tokens of the similar background remains. This verifies our beginning motivation that using a few typical tokens is already enough for learning discriminative features for video representation.

5 CONCLUSION

In this work, we propose a multi-segment clustering algorithm to reduce the number of redundant tokens of continuous video frames, and achieve segment-level alignment of video and text representations for text-video retrieval task. Our method, named CenterCLIP as we only reserve center tokens of token clusters and drop non-center tokens, is based on the knowledge of large-scale image-text pairs pre-trained model – CLIP. We take extensive experiments on four common text-video multi-modal datasets: MSR-VTT, MSVD, LSMDC, and ActivityNet. CenterCLIP achieves state-of-the-art performance on all these four datasets and surpass the old SOTA by a large margin. At the same time, CenterCLIP realizes a decent reduction in memory costs and speedup of inference time.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China under Grant No. 2020AAA0108800. Thanks Naiyuan Liu for his helpful discussions.
