Match4Match: Enhancing Text-Video Retrieval by Maximum Flow with Minimum Cost

Zhongjie Duan
East China Normal University
Shanghai, China
zjduan@stu.ecnu.edu.cn

Chengyu Wang
Alibaba Group
Hangzhou, China
chengyu.wcy@alibaba-inc.com

Cen Chen*
East China Normal University
Shanghai, China
cenchen@dase.ecnu.edu.cn

Wenmeng Zhou
Alibaba Group
Hangzhou, China
wenmeng.zwm@alibaba-inc.com

Jun Huang
Alibaba Group
Hangzhou, China
huangjun.hj@alibaba-inc.com

Weining Qian
East China Normal University
Shanghai, China
wnqian@dase.ecnu.edu.cn

ABSTRACT

With the explosive growth of video and text data on the web, text-video retrieval has become a vital task for online video platforms. Recently, text-video retrieval methods based on pre-trained models have attracted a lot of attention. However, existing methods cannot effectively capture the fine-grained information in videos, and typically suffer from the *hubness problem* where a collection of similar videos are retrieved by a large number of different queries. In this paper, we propose Match4Match, a new text-video retrieval method based on CLIP (Contrastive Language-Image Pretraining) and graph optimization theories. To balance calculation efficiency and model accuracy, Match4Match seamlessly supports three inference modes for different application scenarios. In fast vector retrieval mode, we embed texts and videos in the same space and employ a vector retrieval engine to obtain the top K videos. In fine-grained alignment mode, our method fully utilizes the pre-trained knowledge of the CLIP model to align words with corresponding video frames, and uses the fine-grained information to compute text-video similarity more accurately. In flow-style matching mode, to alleviate the detrimental impact of the hubness problem, we model the retrieval problem as a combinatorial optimization problem and solve it using maximum flow with minimum cost algorithm. To demonstrate the effectiveness of our method, we conduct experiments on five public text-video datasets. The overall performance of our proposed method outperforms state-of-the-art methods. Additionally, we evaluate the computational efficiency of Match4Match. Benefiting from the three flexible inference modes, Match4Match can respond to a large number of query requests with low latency or achieve high recall with acceptable time consumption.

CCS CONCEPTS

• **Information systems** → **Video search**.

*Corresponding author.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583365

KEYWORDS

multimodal learning, video retrieval, network flow

ACM Reference Format:
Zhongjie Duan, Chengyu Wang, Cen Chen, Wenmeng Zhou, Jun Huang, and Weining Qian. 2023. Match4Match: Enhancing Text-Video Retrieval by Maximum Flow with Minimum Cost. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583365

1 INTRODUCTION

Video is one of the most popular communication mediums that is capable of storing rich information. With the rapid growth of online videos and texts on the web, cross-modal learning has become a heated topic in the field of computer vision and natural language processing [30]. In order to help users to find relevant videos efficiently, video retrieval becomes a fundamental and critical task for online video platforms.

Traditional video retrieval methods [1, 57] mainly focus on keyword matching. However, these methods only search videos based on video captions and cannot utilize the rich information from video images. To overcome this pitfall, researchers pay attention to cross-modal learning. The large amount of text and video data on the internet has created favorable conditions for the application of deep learning for video retrieval. The dominant method in recent years is to embed videos and texts into the same vector space. For example, some studies [46, 47] use CNN as video encoder and RNN as text encoder, and use distance metrics of vectors to search for related videos. The retrieval performance relies on the model architecture and training datasets. With the establishment of large-scale datasets and the development of related research, some pre-trained models were proposed [9, 29, 62] for image classification tasks. Subsequently, Collaborative Experts [43] was designed to aggregate information from different pre-trained models, which has been demonstrated to be a strong baseline in text-video retrieval tasks. After Transformer [58] was proposed, Transformer based models became popular in both computer vision [18] and neural language processing [34]. Recently proposed CLIP [51] leverages Transformer and contrastive learning to embed sentences and images into the same space. CLIP was trained on an extremely large text-image dataset and stored abundant knowledge for text-image retrieval. It inspired many following studies. By fine-tuning the CLIP model on text-video datasets, CLIP4CLIP [44] transferred the

knowledge of CLIP to the video retrieval setting, and outperformed the existing methods by a large margin. Several recent works focus on modelling fine-grained information for cross-modal retrieval by aligning the words to the corresponding frames using GCN [12], shared centers [20] or clustering [27, 66]. Different from the methods that directly use distance metrics, these methods typically embed each word and frame into a joint embedding space and then calculate the overall text-video similarities for retrieval. Such methods are able to capture fine-grained semantic differences, thus further improving the task performance, however, more inference overheads are inevitably introduced.

Despite the advancement, we suggest that existing text-video retrieval methods can still be further improved in the following aspects. First, the knowledge stored in the pre-training models is rich and should be fully used for fine-tuning. The majority of CLIP-based methods add additional sub-networks with trainable parameters to the model, which may increase the uncertainty of training. Empirically, the study [38] proved that the fine-tuning method could distort the pre-trained features and influence the performance significantly. Second, the hubness problem is still crucial. There will always be a small number of videos retrieved by a large number of queries. The hubness problem has received attention from researchers, among which QB-NORM [7] was proposed to reduce the adverse impact, but the improvement is minimal. Third, the computing resource requirement for training is very high. Most retrieval methods are based on contrastive learning. In contrastive learning, large batch size is beneficial to improving the performance but requires too much GPU memory [23]. In this paper, we propose a new text-video retrieval method named Match4Match, which supports three flexible inference modes to balance efficiency and accuracy. To reduce the uncertainty of training, we design an architecture that uses the pre-trained CLIP itself for token-frame alignment without any trainable parameters. In order to prevent the hubness problem from affecting the retrieval results, we employ maximum flow and minimum-cost flow algorithms for optimizing retrieval results. To improve the training efficiency, we propose a contrastive gradient accumulation algorithm, which can reduce the computing resource requirement significantly.

We compared our approach with state-of-the-art methods on five public text-video retrieval datasets. Experimental results demonstrate the effectiveness of our method. Although we employ network flow algorithms in Match4Match, the efficiency analysis proves that the time consumption is highly acceptable. In addition, the three inference modes can satisfy different computational resource and accuracy requirements.[1]

The main contributions of our paper include:

- We propose a novel text-video retrieval approach named Match4Match, which seamlessly supports three inference modes to satisfy different computational resource and accuracy requirements.
- Based on graph optimization theory, we design a parameter-free architecture to capture fine-grained token-frame alignment information, and a flow-style matching algorithm to minimize the detrimental impact of hubness problems.
- The experimental results demonstrate that the overall performance of Match4Match outperforms state-of-the-art methods, and the comprehensive efficiency analysis proves that the time consumption is highly acceptable.

[1]Source codes will be released in EasyNLP [59] (https://github.com/alibaba/EasyNLP).

2 RELATED WORK

In this section, we briefly review the related studies of text-video retrieval and network flow algorithms.

2.1 Text-Video Retrieval

In recent years, a large number of prior works have been proposed for cross-modal retrieval tasks. Some existing studies focused on embedding videos and texts into the same space and searching videos using vector queries. In 2016, Otani et al. [46] and Pan et al. [47] used a video encoder and a text encoder to jointly learn the representation of videos and sentences. They demonstrated the feasibility of joint embedding. In 2020, ViT [18] was proposed as a visual model, and outperformed other baselines in many computer vision tasks. Consequently, CLIP [51] employed ViT [18] to embed images and used a Transformer Encoder [58] as a text encoder. CLIP was pre-trained on a large-scale text-image dataset and made great progress in cross-modal research. Following the study of CLIP, Portillo et al. [50] tried to apply this pre-trained model to video retrieval tasks without fine-tuning, and demonstrated the potential of the pre-trained CLIP model. Based on CLIP and contrastive learning, an empirical study [44] proposed a strong baseline using four different similarity definitions. To further improve the performance of CLIP, Chen et al. [12] used GCN [60] for the alignment of words and frames, and CLIP2Video [20] leveraged shared centers for discovering the shared information between videos and texts. CAMoE [14] utilized three expert models for extracting more information from texts and videos. QB-NORM [7] could reduce the influence of hubness problems by normalizing the similarities using a querybank. MDMMT-2 [39] was pre-trained on multiple datasets simultaneously and worked well on downstream tasks that lack large amounts of training data. Inspired by these methods, we propose a novel approach that supports both coarse-grained fast retrieval and fine-grained reranking.

2.2 Network Flow

The maximum flow and minimum-cost flow problems are classic combinatorial graph optimization problems. Over the last several decades, extensive research has been conducted both in theory and application [55]. In the maximum flow and minimum-cost flow problem, given a directed graph where each edge is assigned capacity and cost, we are supposed to find a maximum flow with minimum cost from the source node to the sink node satisfying the capacity limit. In 1951, the first algorithm with pseudo-polynomial time complexity was proposed by Dantzig [16]. After this, Ford-Fulkerson [21], Dinic [17] and Edmonds-Karp [19] algorithms were proposed as faster algorithms. Nowadays the most popular algorithm in practice is Goldberg-Tarjan minimum-cost flow algorithm [26], a cost-scaling push-relabel algorithm. In application, network flow algorithms are widely applied in airline schedule planning [5] and traffic flow management [6]. However, the main drawback of

network flow algorithms is the high time complexity. Several studies focused on improving the efficiency by parallel computation [31] and approximate algorithms [15]. Recently, Chen et al. [11] claimed that they proposed an algorithm in almost linear time. In practice, Google developed a high-efficiency graph optimization framework [49]. These existing studies make it possible to apply network flow algorithms to large-scale text-video retrieval problems.

3 METHODOLOGY

3.1 Overview

In text-video retrieval problems, given N_t sentences $\{t_1, t_2, \ldots, t_{N_t}\}$ and N_v videos $\{v_1, v_2, \ldots, v_{N_v}\}$, we are supposed to find the corresponding video of each sentence. A video v_i is represented by a sequence of frames $\{v_{i,1}, v_{i,2}, \ldots, v_{i,n_v}\}$, and a sentence t_j is represented by a sequence of tokens $\{t_{j,1}, t_{j,2}, \ldots, t_{j,n_t}\}$. Usually, the frames are sampled in the video uniformly, and we follow the same settings as some existing studies [44]. The overview of Match4Match is presented in Figure 2. Match4Match supports three inference modes: fast vector retrieval mode, fine-grained alignment mode and flow-style matching mode. The former is more computationally efficient and the latter can achieve a higher recall. 1) In **fast vector retrieval mode**, we train a coarse-grained model similar to CLIP4CLIP [44]. We embed texts and videos to the same space, and use a vector retrieval engine to obtain top K relevant videos. 2) In **fine-grained alignment mode**, we train a fine-grained model to calculate more accurate token-frame similarities for alignment. The similarities are used to rerank the top K videos. 3) In **flow-style matching mode**, we model the retrieval problem as a combinatorial optimization problem and leverage network flow algorithms to solve it. The flow-style matching can significantly reduce the detrimental influence of the hubness problem.

3.2 Fast Vector Retrieval Mode

Using text-video datasets collected from the Internet, a naive approach is to train a text encoder and a video encoder, and then calculate the similarity between texts and videos. However, a well-trained retrieval model requires too many computing resources, motivating us to develop a retrieval algorithm based on pre-trained models. In this paper, we utilize CLIP [51], an image-language pre-training model. In CLIP, there is a text encoder e_t and a visual encoder e_v trained to convert tokens and frames to embedding vectors in $\mathbb{R}^d$. We use the mean-pooling embedding vector of frames to represent a video, and use the embedding vector of "[EndOfText]" to represent a text, which is a special token at the last of the token sequence, i.e.,

$$\overline{e}_v(v_j) = \frac{1}{n_v}\sum_{l=1}^{n_v} e_v(v_{j,l}), \tag{1}$$

$$\overline{e}_t(t_i) = e_t(t_{i,n_t}). \tag{2}$$

The similarity in CLIP is defined as the cosine similarity. We define the coarse-grained text-video similarity $\overline{S}(t_i, v_j)$ as the cosine similarity between $\overline{e}_t(t_i)$ and $\overline{e}_v(v_j)$:

$$\overline{S}(t_i, v_j) = \frac{\overline{e}_t(t_i)\cdot\overline{e}_v(v_j)}{|\overline{e}_t(t_i)||\overline{e}_v(v_j)|}. \tag{3}$$

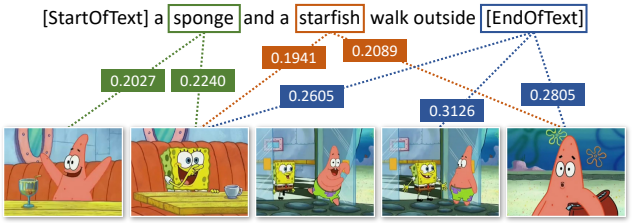

Figure 1: An example of token-frame alignment using pre-trained CLIP. The similarity between a token and its corresponding frame is higher than others.

In order to further transfer the knowledge from text-image tasks to text-video retrieval tasks, we fine-tune the model on downstream text-video datasets. After fine-tuning, we normalize all video embeddings and store them in a database for retrieval. Given a sentence t_i, as shown in Figure 2 (a), we first convert it to a text embedding vector $\overline{e}_t(t_i)$, then find the vectors with high cosine similarity in the database, and finally obtain the corresponding videos $\hat{V}(t_i)$. Some vector retrieval algorithms [32, 56] are utilized for maximum inner product search. These algorithms can search for vectors quickly without scanning all videos.

3.3 Fine-Grained Alignment Mode

The fast vector retrieval mode runs very quickly but is hard to guarantee high accuracy. In fine-grained alignment mode, we first obtain the top K relevant videos $\hat{V}(t_i)$ in fast vector retrieval mode, then calculate the fine-grained similarities $\{S(t_i, v_j)|v_j \in \hat{V}(t_i)\}$ using the fine-grained model to rerank the K videos.

Considering the fact that CLIP is a model pre-trained on a large-scale text-image dataset, we can easily obtain reliable similarities between tokens and frames. In Figure 1, we present an example that shows the similarity between a sentence and its corresponding video. We find that the similarity between a token ("sponge" and "starfish") and the frame in which the corresponding cartoon character occurs is higher than others, even if the model is not fine-tuned on any text-video datasets. The special token "[EndOfText]" represents the whole sentence, thus we can find the key frame where the two characters both occur. According to the similarities, we can align frames to their corresponding tokens and vice versa.

To compute fine-grained text-video similarities, we focus on some representative token-frame alignments. We define the similarity between text t_i and video v_j as:

$$S(t_i, v_j) = \frac{\sum_{(k,l)} f(t_{i,k}, v_{j,l}) s(t_{i,k}, v_{j,l})}{\sum_{(k,l)} f(t_{i,k}, v_{j,l})}, \tag{4}$$

where $f(t_{i,k}, v_{j,l}) \in \{0, 1\}$ represents whether the token-frame pair $(t_{i,k}, v_{j,l})$ is selected, and $s(t_{i,k}, v_{j,l})$ is the token-frame cosine similarity between $e_t(t_{i,k})$ and $e_v(v_{j,l})$.

As we described above, we can extract the alignment information by selecting the token-frame pairs with high similarity. However, directly using a greedy strategy usually leads to an imbalance of choice. Some tokens (frames) may never be selected, thus their embedding will not be updated. This condition will result in sub-optimal training performance. To overcome this pitfall, we restrict

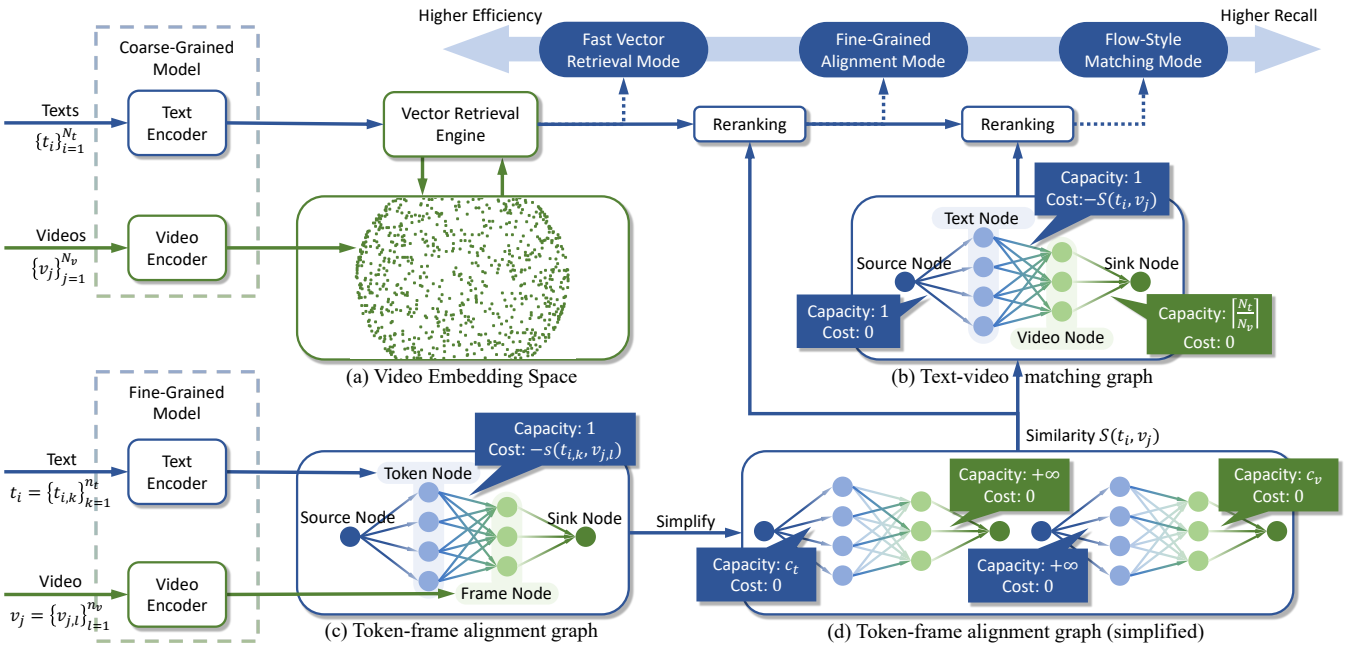

Figure 2: The overview of Match4Match. Match4Match supports three inference modes: fast vector retrieval mode, fine-grained alignment mode and flow-style matching mode. The three inference modes are designed for different computational resources and accuracy requirements, where the former is more efficient and the latter can achieve higher recall.

the number of times that each token (frame) is selected. We model this selection problem as a combinatorial optimization problem:

$$\max_{f} \sum_{(k,l)} s(t_{i,k}, v_{j,l}) f(t_{i,k}, v_{j,l}), \tag{5}$$

$$\text{s.t.} \begin{cases} f(t_{i,k}, v_{j,l}) \in \{0,1\}; & k = 1,2,\ldots,n_t; l = 1,2,\ldots,n_v; \\ \sum_k f(t_{i,k}, v_{j,l}) \le c_v; & l = 1,2,\ldots,n_v; \\ \sum_l f(t_{i,k}, v_{j,l}) \le c_t; & k = 1,2,\ldots,n_t. \end{cases} \tag{6}$$

To calculate the optimal solution, we leverage network flow optimization theories. As shown in Figure 2 (c), a directed graph is constructed to represent this optimization problem. There are $n_t + n_v + 2$ nodes, where token node $u(t_{i,k})$ represents the token $t_{i,k}$, frame node $u(v_{j,l})$ represents the frame $v_{j,l}$, and the other two extra nodes represent the source node and sink node. We assign capacity and cost to each directed edge (arc). The capacity of an edge is c_t if it starts from the source node, and is c_v if it ends with the sink node. With these settings, we can make sure each token is selected at most c_t times, and each frame is selected at most c_v times. If an edge starts from a token node $u(t_{i,k})$ and ends with a frame node $u(v_{j,l})$, its capacity is 1 and its cost is $-s(t_{i,k}, v_{j,l})$. The cost of other edges is 0. Therefore, we can find the optimal solution of this optimization problem by finding a maximum flow f with minimum cost on the graph.

During the training procedure, the token-frame similarities are updated by gradient-based optimization algorithms, thus the graph is also updated dynamically. We need to recalculate the flow frequently, which makes the training procedure slow. In order to improve efficiency, we simplify the optimization problem to find a sub-optimal solution that can be solved quickly. As shown in Figure 2 (d), we split the graph structure into two dual parts. In the first part, the capacities of edges that end with the sink node are unlimited. In the second part, the capacities of edges that start with the source node are unlimited. In other words, we align each token to its corresponding frames in the first part, and align each frame to its corresponding tokens in the second part. Using f_A and f_B to denote the minimum-cost flow of the two parts, we obtain two text-video similarities:

$$S_A(t_i, v_j) = \frac{\sum_{(k,l)} f_A(t_{i,k}, v_{j,l}) s(t_{i,k}, v_{j,l})}{\sum_{(k,l)} f_A(t_{i,k}, v_{j,l})}, \tag{7}$$

$$S_B(t_i, v_j) = \frac{\sum_{(k,l)} f_B(t_{i,k}, v_{j,l}) s(t_{i,k}, v_{j,l})}{\sum_{(k,l)} f_B(t_{i,k}, v_{j,l})}. \tag{8}$$

Neither $s_A(t_i, v_j)$ nor $s_B(t_i, v_j)$ satisfies reflexivity, but we can integrate the two similarities together:

$$S(t_i, v_j) = \frac{S_A(t_i, v_j) + S_B(t_i, v_j)}{2}. \tag{9}$$

Now $S(t_i, v_j) = S(v_j, t_i)$. Obviously, if we set $c_t = c_v = 1$, the text-video similarity will be computed easily:

$$S(t_i, v_j) = \frac{1}{2}\left(\frac{1}{n_t}\sum_k \max_l s(t_{i,k}, v_{j,l}) + \frac{1}{n_v}\sum_l \max_k s(t_{i,k}, v_{j,l})\right). \tag{10}$$

This simplified similarity aligns the frame with the highest similarity to the token and vice versa.

3.4 Flow-Style Matching Mode

In flow-style matching mode, we further improve the retrieval performance. In the above two inference modes and most existing CLIP-based text-video retrieval methods, the most relevant video $\hat{v}(t_i)$ is selected by an arg max function:

$$\hat{v}(t_i) = \arg\max_{v_j} S(t_i, v_j). \tag{11}$$

This retrieval strategy is simple and easy to be implemented, but it suffers significantly from the longstanding hubness problem [52]. Similar to token-frame alignment, we model the text-to-video retrieval problem as another combinatorial optimization problem, formulated as:

$$\max_{f} \sum_{(i,j)} S(t_i, v_j) f(t_i, v_j), \tag{12}$$

$$\text{s.t.} \begin{cases} f(t_i, v_j) \in \{0, 1\}; & i = 1, 2, \ldots, N_t; j = 1, 2, \ldots, N_v; \\ \sum_j f(t_i, v_j) \le 1; & i = 1, 2, \ldots, N_t; \\ \sum_i f(t_i, v_j) \le \lceil \frac{N_t}{N_v} \rceil; & j = 1, 2, \ldots, N_v. \end{cases} \tag{13}$$

In this optimization problem, we limit the number of times that each video is matched to $\lceil \frac{N_t}{N_v} \rceil$. This limitation can directly reduce the adverse effect caused by the hubness problem. If $N_t \le N_v$, this optimization problem will be a maximum weight matching problem on the bipartite graph, and we can utilize maximum weight matching algorithms (e.g., Kuhn-Munkres Algorithm [37]) to obtain the optimal solution. If $N_t > N_v$, a video will match more than one text. Employing graph optimization theory again, we construct a new graph. In Figure 2 (b), the new graph is similar to that in Figure 2 (c). There are N_t text nodes $u(t_1), u(t_2), \ldots, u(t_{N_t})$, N_v video nodes $u(v_1), u(v_2), \ldots, u(v_{N_v})$, a source node u_{source} and a sink node u_{sink} in the graph. This graph is sparse and only contains $N_t + N_t K + N_v$ edges. The capacity of an edge is 1 if it starts from the source node, and is $\lceil \frac{N_t}{N_v} \rceil$ if it ends with the sink node. If an edge starts from a token node $u(t_i)$ and ends with a frame node $u(v_j)$, its capacity is 1 and its cost is $-S(t_i, v_j)$. Obviously, the maximum flow is at most N_t because of the capacity limitation that starts from the source node, thus the algorithm aligns at most one video to every text. The minimum cost corresponds to the maximum sum of similarities. The matched video of each text t_i is the video v_j that satisfies $f(t_i, v_j) = 1$.

Another important aspect to examine is the efficiency of inference. For Goldberg-Tarjan minimum-cost flow algorithm [26], in the worst cases, the time complexity is up to $O(N^2 M \log(NC))$, where N is the number of nodes, M is the number of edges and C is the maximum absolute value of edge costs. Although the time complexity seems high, in this problem, the graph architecture is very sparse and randomized, therefore the minimum-cost flow algorithm runs fast. Benefiting from the studies of efficient implementation [8, 25], the algorithm can run in almost linear time.

To combine the flow-style matching results with other existing methods, we first replace the similarity with the weighted sum of fine-grained similarity and the optimal flow:

$$S_f(t_i, v_j) = S(t_i, v_j) + \beta f(t_i, v_j), \tag{14}$$

where the hyperparameter β controls the weight of flow-style matching results. Next, apply Dual Softmax function [14] to the similarities. Note that Dual Softmax function requires all $N_t \times N_v$ similarity values, which is very time-consuming. To improve the efficiency, we only apply it to the top K relevant videos, i.e., recalculating the $N_t \times K$ similarity values:

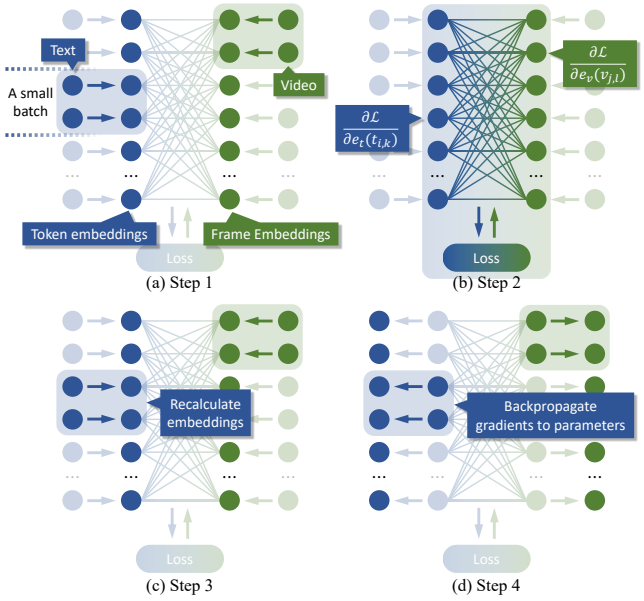

Figure 3: The schematic diagram of contrastive gradient accumulation algorithm.

$$S_{\text{F}}^{\text{v2t}}(t_i, v_j) = \frac{\exp(\alpha S_f(t_i, v_j))}{\sum_{k=1}^{N_t} I(v_j \in \hat{V}(t_k)) \exp(\alpha S_f(t_k, v_j))}, \tag{15}$$

$$S_{\text{F}}^{\text{t2v}}(t_i, v_j) = \frac{\exp(\alpha S_f(t_i, v_j))}{\sum_{k=1}^{N_v} I(v_k \in \hat{V}(t_i)) \exp(\alpha S_f(t_i, v_k))}, \tag{16}$$

$$S_{\text{F}}(t_i, v_j) = S_{\text{F}}^{\text{v2t}}(t_i, v_j) \cdot S_{\text{F}}^{\text{t2v}}(t_i, v_j), v_j \in \hat{V}(t_i), \tag{17}$$

where I is an identity function. $I(x) = 1$ if x is true and $I(x) = 0$ otherwise. Finally, we rerank the videos according to $S_{\text{F}}(t_i, v_j)$.

3.5 Training Algorithm

To train the model, we employ symmetric cross entropy loss, which is based on InfoNCE loss [45] and widely used in other retrieval models [4, 27, 44]. This loss function is calculated over a batch of text-video pairs $\{(t_1, v_1), (t_2, v_2), \ldots, (t_B, v_B)\}$:

$$\mathcal{L}_{\text{t2v}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\tau S(t_i, v_i))}{\sum_{j \ne i} \exp(\tau S(t_i, v_j))}, \tag{18}$$

$$\mathcal{L}_{\text{v2t}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\tau S(t_i, v_i))}{\sum_{j \ne i} \exp(\tau S(t_j, v_i))}, \tag{19}$$

$$\mathcal{L} = \mathcal{L}_{\text{t2v}} + \mathcal{L}_{\text{v2t}}. \tag{20}$$

This loss function is compatible with both coarse-grained and fine-grained models, differing only in the definition of similarity. Each text t_i is compared to all the B videos in the batch. Consequently, the batch size should be large enough [13] to improve performance. We need more GPU memory to increase the batch size. Besides using more resources for parallel computing, we are

interested in designing an algorithm with low GPU memory requirements. Wu et al. [61] proposed memory bank, He et al. [28] proposed a momentum updating method, and Liu et al. [42] introduced them into text-video retrieval problems. These methods extend the negative sample set by storing the embedding vectors during the training procedure, but essentially the stored information is not from the updated model. This inconsistency may lead to sub-optimal performance. Inspired by gradient accumulation, we design a contrastive gradient accumulation algorithm.

The contrastive gradient accumulation algorithm is shown in Figure 3. The original gradient accumulation method cannot be utilized directly because the loss of each sample is not calculated individually. We devise a four-step gradient calculation method. Before calculation, divide the entire batch into B/B' small batches uniformly. Ensure that the GPU device has enough memory to process B' samples. First, calculate the token embeddings and frame embeddings for every small batch. Second, calculate text-video similarities and the loss function to obtain $\frac{\partial \mathcal{L}}{\partial e_t(t_{i,k})}$ and $\frac{\partial \mathcal{L}}{\partial e_v(v_{j,l})}$ for every token and frame. Third, for every small batch, recalculate the embeddings to acquire the intermediate variables required in backpropagation, and then backpropagate the gradient to all parameters. Finally, update the parameters using gradient-based optimization algorithms (e.g., SGD [53] and Adam [35]). Theoretically, the gradient obtained by this algorithm is exactly the same as that without this algorithm. Even though this algorithm increases the time consumption somewhat, it decreases the GPU memory requirement significantly.

4 EXPERIMENTS

To demonstrate the effectiveness of our proposed model, we compare it with state-of-the-art methods on five text-video datasets and analyze its efficiency.

4.1 Datasets and Evaluation Metrics

The experiments are conducted on five datasets: 1) **MSR-VTT** [63] contains 10,000 videos, each with 20 video descriptions. The videos are annotated by Amazon Mechanical Turk. We follow the standard split of MSR-VTT-1kA [22]. 2) **MSVD** [10] consists of 1,970 video snippets with captions. This dataset was collected by Microsoft in 2010. We evaluate our model on the standard split. 3) **LSMDC** [54] is a large scale movie description dataset. This dataset contains 118,081 short video clips extracted from 202 movies. Each video clip has a caption. The standard split is adopted in our experiments. 4) **ActivityNet** [36] is a dataset for human activity understanding. It consists of 20,000 videos. We use the version released in 2016 and follow the settings of [22]. 5) **DiDeMo** [3] is a diverse dataset that contains 10,000 videos collected from Flickr. Each video has a descriptive paragraph. We follow the split from [43].

Following existing studies [4, 44], we use the following five metrics to evaluate the performance of our method both on text-to-video and video-to-text retrieval tasks: 1) **R@K** is the recall at rank K, which is equal to the percentage of queries that contain at least one ground-truth video (text) in the predicted top K results. We set K to 1, 5 and 10. 2) **MdR** is the median rank of the ground-truth results. If there is more than one ground-truth video (text), we use the video (text) with the highest similarity. 3) **MnR** is the mean rank of the ground-truth results.

4.2 Implementation Details

The backbone of our model architecture is CLIP (ViT-L/16). The text encoder is Transformer Encoder [58], and the video encoder is ViT [18]. We initialize the model with the pre-trained weight of CLIP. For more details about the model architecture and the pre-training, please refer to the original paper [51]. To reduce the uncertainty during training, no trainable parameters are added to the model. Each sentence is converted to a token sequence of length 77, which is the default text length of CLIP. For MSR-VTT, MSVD and LSMDC, we uniformly sample 12 frames in each video. For ActivityNet and DiDeMo, the videos are longer than in the above three datasets, so we consider them as paragraph-video retrieval tasks and uniformly sample 64 frames per video. To train the models efficiently, we use Adam as the optimization algorithm. The learning rate is set to 1×10^{-5} and ϵ is set to 1×10^{-3}. The batch size is 64. The contrastive gradient accumulation algorithm mentioned in Section 3.5 is applied to ActivityNet and DiDeMo, where the parameter B' is set to 8. The parameter τ in the loss function is 1×10^2. Considering the consistency during training and inference, we set $\alpha = \tau$ and $\beta = 1$ in the Dual Softmax function. For every downstream dataset, we fine-tune the coarse-grained model and the fine-grained model respectively using only one NVIDIA A100 GPU. The model is trained at most 5 epochs. To achieve the best performance, we use the flow-style matching mode and set K to N_v. We use PyTorch [48] to implement the training algorithm, use OR-Tools [49] to construct the network flow algorithm, and use Faiss [33] as the vector retrieval engine.

4.3 Performance Comparison

The experimental results of Match4Match and other state-of-the-art methods are presented in Table 1, where the results of baseline methods are collected from the articles. The overall performance of CLIP-based approaches (CLIP-Straight, CLIP4CLIP, CLIP2Video, CenterCLIP, CAMoE, MDMMT-2 and CLIP2TV) is better than others. Our method outperforms the existing methods on four datasets. On MSR-VTT and LSMDC, we observed that Match4Match achieves the performance of state-of-the-art approaches. For MSVD, MDMMT-2 is the only one that outperforms our method on this dataset. MSVD only contains 1,970 video snippets, so our model cannot learn too much knowledge by fine-tuning on MSVD. Since MDMMT-2 is pre-trained on additional datasets, it has learned more information than our model before fine-tuning on MSVD. In the absence of additional training datasets, our model can still outperform others. On ActivityNet and DiDeMo, the performance of Match4Match at R@1 is 61.7 and 55.4, exceeding state-of-the-art methods by 10.7 and 11.6, respectively. In our opinion, the main improvement comes from the fine-grained information extracted from tokens and frames. These significant improvements demonstrate the effectiveness of our approach on long videos. We also calculate the t-test on R@1 of Match4Match and the best existing method. The p-value is 0.10, indicating that the t-test rejects the null hypothesis at the significance level of 10%, in favor of the alternative hypothesis that Match4Match reaches higher R@1 than others.

Table 1: The performance of Match4Match and other competing methods on five datasets. The best results are bolded, and the second best are underscored.

Dataset	Method	Text-to-video					Video-to-text				
		R@1 ↑	R@5 ↑	R@10 ↑	MdR ↓	MnR ↓	R@1 ↑	R@5 ↑	R@10 ↑	MdR ↓	MnR ↓
MSR-VTT	JSFusion [65]	10.2	31.2	43.2	13	-	-	-	-	-	-
	Collaborative Experts [43]	20.9	48.8	62.4	6	28.2	20.6	50.3	64.0	5.3	25.1
	TACo [64]	28.4	57.8	71.2	4	-	-	-	-	-	-
	ALPRO [41]	33.9	60.7	73.2	3	-	-	-	-	-	-
	CLIPBERT [40]	22.0	46.8	59.9	6	-	-	-	-	-	-
	FCA-Net [27]	23.2	55.6	70.3	3	-	-	-	-	-	-
	MMT [22]	26.6	57.1	69.6	4	24.0	27.0	57.5	69.7	3.7	21.3
	CLIP-Straight [50]	31.2	53.7	64.2	4	-	27.2	51.7	62.6	5	-
	Frozen [4]	31.0	59.5	70.5	3	-	-	-	-	-	-
	CLIP4CLIP [44]	44.5	71.4	81.6	2	15.3	42.7	70.9	80.6	2	11.6
	CLIP2Video (+QB-NORM) [7]	47.2	73.0	83.0	2	-	-	-	-	-	-
	CAMoE [14]	47.3	74.2	84.5	2	11.9	49.1	74.3	84.3	2	9.9
	MDMMT-2 [39]	48.5	75.4	83.9	2	13.8	-	-	-	-	-
	CLIP2TV [24]	52.9	**78.5**	86.5	**1**	12.8	54.1	**77.4**	**85.7**	**1**	9.0
	Match4Match (Ours)	**55.5**	77.8	**86.6**	**1**	**9.8**	**55.7**	76.5	84.7	**1**	**8.4**
MSVD	Collaborative Experts [43]	19.8	49.0	63.8	6	23.1	-	-	-	-	-
	SSML [2]	20.3	49.0	63.3	6	-	-	-	-	-	-
	CLIP-Straight [50]	37.0	64.1	73.8	3	-	59.9	85.2	90.7	**1**	-
	Frozen [4]	33.7	64.7	76.3	3	-	-	-	-	-	-
	CLIP4CLIP [44]	46.2	76.1	84.6	2	10.0	62.0	87.3	92.6	**1**	4.3
	CLIP2Video (+QB-NORM) [7]	48.0	77.9	86.2	2	-	-	-	-	-	-
	CenterCLIP [66]	50.6	80.3	88.4	**1**	**8.4**	68.4	90.1	95.0	**1**	**3.0**
	CAMoE [14]	49.8	79.2	87.0	-	9.4	-	-	-	-	-
	MDMMT-2 [39]	**56.8**	**83.1**	**89.2**	**1**	8.8	-	-	-	-	-
	Match4Match (Ours)	53.2	81.2	88.6	**1**	8.7	**70.4**	**91.3**	**95.4**	**1**	3.8
LSMDC	JSFusion [65]	9.1	21.2	34.1	36	-	-	-	-	-	-
	Collaborative Experts [43]	11.2	26.9	34.8	25.3	-	-	-	-	-	-
	MMT [22]	12.9	29.9	40.1	19.3	75.0	12.3	28.6	38.9	20	76.0
	CLIP-Straight [50]	11.3	22.7	29.2	56.5	-	6.8	16.4	22.1	73	-
	Frozen [4]	15.0	30.8	39.8	20	-	-	-	-	-	-
	CLIP4CLIP [44]	21.6	41.8	49.8	11	58.0	20.9	40.7	49.1	11	53.9
	CLIP4CLIP (+QB-NORM) [7]	22.4	40.1	49.5	11	-	-	-	-	-	-
	CenterCLIP [66]	24.2	46.2	55.9	8	**47.3**	24.5	46.4	55.8	**7**	41.3
	CAMoE [14]	25.9	46.1	53.7	-	54.4	-	-	-	-	-
	MDMMT-2 [39]	26.9	46.7	55.9	**6.7**	48.0	-	-	-	-	-
	Match4Match (Ours)	**27.9**	**47.0**	**56.6**	7	47.9	**29.0**	**47.6**	**56.7**	**7**	**40.0**
ActivityNet	Collaborative Experts [43]	18.2	47.7	-	6	23.1	17.7	46.6	-	6	24.4
	TACo [64]	30.4	61.2	-	3	-	-	-	-	-	-
	CLIPBERT [40]	21.3	49.0	63.5	6	-	-	-	-	-	-
	MMT [22]	28.7	61.4	-	3.3	16.0	28.9	61.1	-	4	17.1
	CLIP4CLIP [44]	40.5	72.4	-	2	7.4	42.5	74.1	85.8	2	6.6
	CenterCLIP [66]	46.2	77.0	87.6	2	5.7	46.7	77.1	88.0	2	5.5
	CAMoE [14]	51.0	77.7	-	-	-	49.9	77.4	-	-	-
	Match4Match (Ours)	**61.7**	**82.9**	**90.7**	**1**	**4.6**	**61.7**	**82.9**	**90.3**	**1**	**4.4**
DiDeMo	Collaborative Experts [43]	16.1	41.1	-	8.3	43.7	15.6	40.9	-	8.2	42.4
	ALPRO [41]	35.9	67.5	78.8	3	-	-	-	-	-	-
	CLIPBERT [40]	20.4	48.0	60.8	6	-	-	-	-	-	-
	Frozen [4]	34.6	65.0	74.7	3	-	-	-	-	-	-
	CLIP4CLIP [44]	43.4	70.2	80.6	2	17.5	42.5	70.6	80.2	2	11.6
	CLIP4CLIP (+QB-NORM) [7]	43.5	71.4	80.9	2	-	-	-	-	-	-
	CAMoE [14]	43.8	71.4	-	-	-	45.5	71.2	-	-	-
	Match4Match (Ours)	**55.4**	**79.1**	**85.5**	**1**	**10.8**	**55.4**	**78.8**	**86.0**	**1**	**6.6**

Table 2: The time consumption of Match4Match on MSR-VTT and LSMDC*, where M1 denotes fast vector retrieval mode, M2 denotes fine-grained alignment mode, and M3 denotes flow-style matching mode.

Dataset	Inference mode	Time consumption (millisecond)	Percentage of time consumption in each part									
			Coarse-grained		Fine-grained			Vector retrieval engine		Others		
			Video encoding	Text encoding	Video encoding	Text encoding	Similarity calculation	Training	Inference	Flow-style matching	Sparse Dual Softmax	Reranking
MSR-VTT	Offline	25014	53.44%	-	46.50%	-	-	0.06%	-	-	-	-
	M1	233	-	84.98%	-	-	-	-	15.02%	-	-	-
	M2	487	-	40.66%	-	42.09%	8.62%	-	7.19%	-	-	1.44%
	M3	575	-	34.43%	-	35.65%	7.30%	-	6.09%	13.74%	1.57%	1.22%
LSMDC*	Offline	2783165	52.20%	-	44.49%	-	-	3.31%	-	-	-	-
	M1	25132	-	79.89%	-	-	-	-	20.11%	-	-	-
	M2	52530	-	38.22%	-	42.53%	9.58%	-	9.62%	-	-	0.04%
	M3	53296	-	37.67%	-	41.92%	9.45%	-	9.48%	1.01%	0.43%	0.04%

Table 3: The performance of Match4Match on MSR-VTT for text-to-video retrieval

K	Mode	R@1	R@5	R@10	MdR	MnR
30	Fast vector retrieval	45.1	69.1	81.5	2	[6.7, 90.0]
	Fine-grained alignment	50.0	73.4	83.3	2	[6.0, 89.3]
	Flow-style matching	53.6	74.4	83.4	1	[5.9, 89.2]

4.4 Inference Efficiency Analysis

To evaluate the inference efficiency and explore the contribution of each component, we conduct experiments on two datasets. Table 2 presents the inference time with respect to different modes and components. Meanwhile, Table 3 shows the performance of each mode, evaluated on a test set of 1,000 text-video pairs for MSR-VTT with the setting $K = 30$. Considering the error of the running time, we run it three times and report the average value. Note that we cannot obtain an accurate MnR because the model only outputs the top K relevant videos instead of ranking all videos. We report the possible range of MnR in Table 3. To simulate real application scenarios with a large number of videos, we construct a new dataset LSMDC*, which consists of 100,000 text-video pairs sampled from the training set of LSMDC, and evaluate the inference time on this dataset.

In Table 2, we observe that the most time-consuming part is the two video encoding processes. For a text-to-video retrieval system, these processes can be done offline. In fast vector retrieval mode, we only need to perform coarse-grained text encoding and vector retrieval engine inference online, where the former requires more time than the latter. For a single query, we need approximately 0.23 milliseconds to return the top 30 relevant videos in MSR-VTT, and 0.25 milliseconds in LSMDC*. In fine-grained alignment mode, in addition to the computation in fast vector retrieval mode, we need to calculate the fine-grained similarities and rerank the videos. The time consumption in fine-grained alignment mode is about twice that in fast vector retrieval mode. In flow-style matching mode, we need more time to compute the minimum-cost flow and apply Dual Softmax function to the similarities. Benefiting from the high-efficiency implementation of Goldberg-Tarjan minimum-cost flow algorithm provided by OR-Tools [49], this component is capable of processing 100,000 videos quickly. In Table 3, we can see that the performance in flow-style matching mode is still high enough when $K = 30$. The performance of fast vector retrieval mode is lower than the other two modes, but it is the fastest mode. The three inference modes are designed for different computational resources and accuracy requirements.

4.5 Training Efficiency Analysis

We also evaluate the effects of the contrastive gradient accumulation algorithm. Considering the resource requirement of one epoch when the batch size $B = 64$, for MSR-VTT, we need 74GB GPU memory and 1.5 hours of computation time without it. While using this algorithm, we need 12GB GPU memory and 2.2 hours. For ActivityNet, following the same settings, the training program without it needs nearly 370GB GPU memory due to the longer video length, thus more GPU devices are required. But we only need 50GB while using it. This algorithm dramatically decreases the GPU memory requirements with an acceptable time penalty.

5 CONCLUSION AND FUTURE WORK

In this paper, we investigate the text-video retrieval approaches. Specifically, we propose a novel method (Match4Match) that supports three inference modes. The three inference modes are designed for different computational resources and accuracy requirements. In fast vector retrieval mode, we can obtain the top K relevant videos quickly using a vector retrieval engine. In fine-grained alignment mode, Match4Match can fully leverage the pre-trained knowledge to capture fine-grained information. In flow-style matching mode, our approach can further improve performance by utilizing network flow algorithms. The overall performance of our method exceeds existing methods. We also design a contrastive gradient accumulation algorithm, making it possible to train the model with large batch size and low GPU memory requirements. In future work, to further enhance the model performance, we will focus on training models more effectively on large-scale datasets.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under grant number 62202170 and Alibaba Group through the Alibaba Innovation Research Program.

REFERENCES

[1] Arnon Amir, Janne Argillander, Murray Campbell, Alexander Haubold, Giridharan Iyengar, Shahram Ebadollahi, Feng Kang, Milind R Naphade, Apostol Natsev, John R Smith, et al. 2003. IBM Research TRECVID-2003 Video Retrieval System.. In *TRECVID*.

[2] Elad Amrani, Rami Ben-Ari, Daniel Rotman, and Alex Bronstein. 2021. Noise estimation using density estimation for self-supervised multimodal learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6644–6652.

[3] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*. 5803–5812.

[4] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1728–1738.

[5] Cynthia Barnhart and Amy Cohn. 2004. Airline schedule planning: Accomplishments and opportunities. *Manufacturing & service operations management* 6, 1 (2004), 3–22.

[6] Dimitris Bertsimas and Sarah Stock Patterson. 2000. The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. *Transportation Science* 34, 3 (2000), 239–255.

[7] Simion-Vlad Bogolin, Ioana Croitoru, Hailin Jin, Yang Liu, and Samuel Albanie. 2022. Cross Modal Retrieval with Querybank Normalisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5194–5205.

[8] Ursula Bünnagel, Bernhard Korte, and Jens Vygen. 1998. Efficient implementation of the Goldberg–Tarjan minimum-cost flow algorithm. *Optimization Methods and Software* 10, 2 (1998), 157–174.

[9] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6299–6308.

[10] David Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 190–200.

[11] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 2022. Maximum flow and minimum-cost flow in almost-linear time. *arXiv preprint arXiv:2203.00671* (2022).

[12] Shizhe Chen, Yida Zhao, Qin Jin, and Qi Wu. 2020. Fine-grained video-text retrieval with hierarchical graph reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10638–10647.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[14] Xing Cheng, Hezheng Lin, Xiangyu Wu, Fan Yang, and Dong Shen. 2021. Improving video-text retrieval by multi-stream corpus alignment and dual softmax loss. *arXiv preprint arXiv:2109.04290* (2021).

[15] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. 2011. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 273–282.

[16] George B Dantzig. 1951. Application of the simplex method to a transportation problem. *Activity analysis and production and allocation* (1951).

[17] Yefim A Dinitz. 1970. An algorithm for the solution of the problem of maximal flow in a network with power estimation. In *Doklady Akademii nauk*, Vol. 194. Russian Academy of Sciences, 754–757.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[19] Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19, 2 (1972), 248–264.

[20] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. 2021. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097* (2021).

[21] Lester Randolph Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* 8 (1956), 399–404.

[22] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. 2020. Multi-modal transformer for video retrieval. In *European Conference on Computer Vision*. Springer, 214–229.

[23] Yanjie Gao, Yu Liu, Hongyu Zhang, Zhengxian Li, Yonghao Zhu, Haoxiang Lin, and Mao Yang. 2020. Estimating gpu memory consumption of deep learning models. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1342–1352.

[24] Zijian Gao, Jingyu Liu, Sheng Chen, Dedan Chang, Hao Zhang, and Jinwei Yuan. 2021. Clip2tv: An empirical study on transformer-based methods for video-text retrieval. *arXiv preprint arXiv:2111.05610* (2021).

[25] Andrew V Goldberg. 1997. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of algorithms* 22, 1 (1997), 1–29.

[26] Andrew V Goldberg and Robert E Tarjan. 1990. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research* 15, 3 (1990), 430–466.

[27] Ning Han, Jingjing Chen, Guangyi Xiao, Hao Zhang, Yawen Zeng, and Hao Chen. 2021. Fine-grained cross-modal alignment network for text-video retrieval. In *Proceedings of the 29th ACM International Conference on Multimedia*. 3826–3834.

[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[29] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[30] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. 2011. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41, 6 (2011), 797–819.

[31] Hiroshi Imai and Kazuo Iwano. 1990. Efficient sequential and parallel algorithms for planar minimum cost flow. In *International Symposium on Algorithms*. Springer, 21–30.

[32] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[33] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[34] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*. 4171–4186.

[35] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[36] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*. 706–715.

[37] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[38] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054* (2022).

[39] Alexander Kunitsyn, Maksim Kalashnikov, Maksim Dzabraev, and Andrei Ivaniuta. 2022. MDMMT-2: Multidomain Multimodal Transformer for Video Retrieval, One More Step Towards Generalization. *arXiv preprint arXiv:2203.07086* (2022).

[40] Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. 2021. Less is more: Clipbert for video-and-language learning via sparse sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7331–7341.

[41] Dongxu Li, Junnan Li, Hongdong Li, Juan Carlos Niebles, and Steven CH Hoi. 2022. Align and Prompt: Video-and-Language Pre-training with Entity Prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4953–4963.

[42] Song Liu, Haoqi Fan, Shengsheng Qian, Yiru Chen, Wenkui Ding, and Zhongyuan Wang. 2021. Hit: Hierarchical transformer with momentum contrast for video-text retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11915–11925.

[43] Yang Liu, Samuel Albanie, Arsha Nagrani, and Andrew Zisserman. 2019. Use what you have: Video retrieval using representations from collaborative experts. *arXiv preprint arXiv:1907.13487* (2019).

[44] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. 2022. CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval and Captioning. *Neurocomputing* (2022).

[45] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[46] Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. 2016. Learning joint representations of videos and sentences with web image search. In *European Conference on Computer Vision*. Springer, 651–667.

[47] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016. Jointly modeling embedding and translation to bridge video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4594–4602.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[49] Laurent Perron and Vincent Furnon. 2022. *OR-Tools*. Google. https://developers.google.com/optimization/

[50] Jesús Andrés Portillo-Quintero, José Carlos Ortiz-Bayliss, and Hugo Terashima-Marín. 2021. A straightforward framework for video retrieval using clip. In *Mexican Conference on Pattern Recognition*. Springer, 3–12.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 8748–8763.

[52] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research* 11, sept (2010), 2487–2531.
[53] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.
[54] Anna Rohrbach, Marcus Rohrbach, and Bernt Schiele. 2015. The long-short story of movie description. In *German conference on pattern recognition*. Springer, 209–221.
[55] Maiko Shigeno. 2004. A SURVEY OF COMBINATORIAL MAXIMUM FLOW ALGORITHMS ON A NETWORK WITH GAINS (< Special Issue> Network Design, Control and Optimization). *Journal of the Operations Research Society of Japan* 47, 4 (2004), 244–264.
[56] Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, Vol. 3. IEEE Computer Society, 1470–1470.
[57] Cees GM Snoek, Marcel Worring, et al. 2009. Concept-based video retrieval. *Foundations and Trends® in Information Retrieval* 2, 4 (2009), 215–322.
[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[59] Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. EasyNLP: A Comprehensive and Easy-to-use Toolkit for Natural Language Processing. (2022). https://doi.org/10.48550/ARXIV.2205.00258
[60] Max Welling and Thomas N Kipf. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.
[61] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3733–3742.
[62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.
[63] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5288–5296.
[64] Jianwei Yang, Yonatan Bisk, and Jianfeng Gao. 2021. Taco: Token-aware cascade contrastive learning for video-text alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11562–11572.
[65] Youngjae Yu, Jongseok Kim, and Gunhee Kim. 2018. A joint sequence fusion model for video question answering and retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 471–487.
[66] Shuai Zhao, Linchao Zhu, Xiaohan Wang, and Yi Yang. 2022. CenterCLIP: Token Clustering for Efficient Text-Video Retrieval. *arXiv preprint arXiv:2205.00823* (2022).

A CASE STUDY

We present an example in MSR-VTT. Given a text query "a soccer team walking outside on the field", Match4Match first output the top K relevant videos using the vector retrieval engine. The top three video are shown in Figure 4. All three videos show a soccer team on the field, where video 3 is the ground-truth video. In video 1, they are celebrating something. In video 2, they are fighting. In video 3, they first go to the field and then begin the soccer match. Only the second and the third frame in video 3 are related to "walking", making the task challenging. In fast vector retrieval mode, the coarse-grained can find the three videos related to the keyword "soccer" but cannot guarantee high accuracy. In fine-grained alignment mode, we rerank these videos using the fine-grained model and video 3 becomes the top 1 video. The token-frame similarities between the text query and video 3 are shown in Figure 5. After fine-tuning, the similarities of the keyword ("soccer") and keyframes (the second frame and the third frame) are highlighted. Most tokens are aligned to the keyframes and most frames are aligned to the keyword, thus the text-video similarity is more accurate than that in fast vector retrieval mode. This example shows how our model extracts fine-grained information.

Figure 4: The top 3 videos retrieved by fast retrieval mode when the model is given a text query "a soccer team walking outside on the field". Video 3 is the ground-truth video.

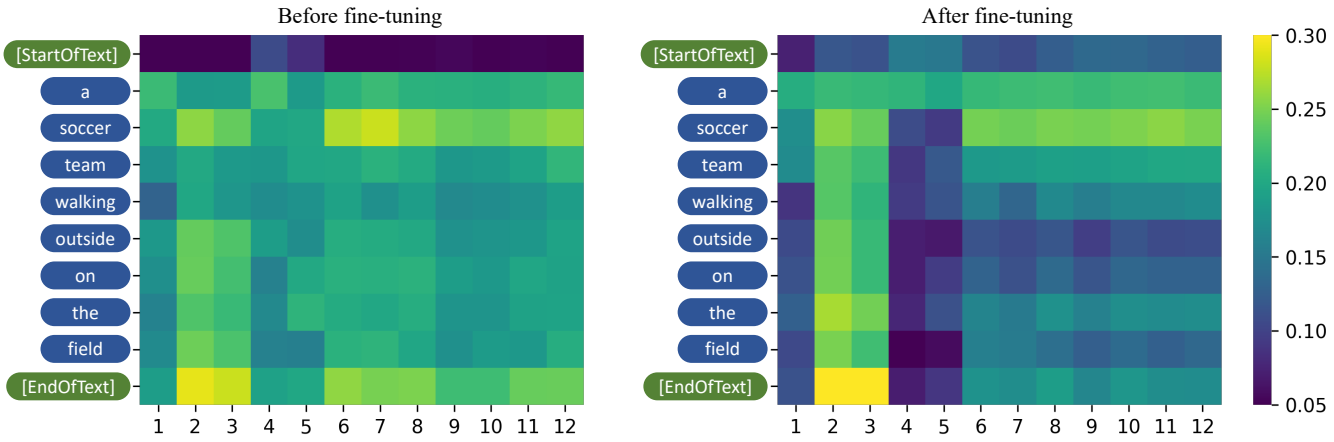

Figure 5: The token-frame similarities between the text query and video 3 in fine-grained alignment mode.