
Fine-tuning CLIP4Clip: Efficiency and Optimization for Video-Text Retrieval

Sophia Wang, Xiaoyang Ma, Hang Ruan, Sijia Ma
Carnegie Mellon University
sophiaw3, xm3, hruan, sijiam@andrew.cmu.edu

Abstract

Video understanding poses significant challenges in effectively capturing the intricate relationships between visual content and textual descriptions. Despite achieving state-of-the-art results on MSRVT and similar benchmarks, CLIP4Clip, a video-text retrieval model based on OpenAI’s pre-trained CLIP model, incurs heavy computational cost. This limits its accessibility outside well-funded labs and rules out low-power or real-time deployment. In this project, we address the gap by investigating model compression and efficient attention mechanisms to reduce both training time and inference latency. We also systematically explore algorithmic and architectural optimizations—such as Spatial–Temporal Auxiliary Network, layer reduction, frame selection, dynamic sequence lengths, and mixed-precision—to reduce computation while maintaining comparable accuracy on standard benchmarks.

1 Introduction

Video–text retrieval aligns natural-language queries with corresponding video segments and underpins applications from content search to automated captioning. Building on the success of CLIP’s image–text pre-training (1), CLIP4Clip adapts these representations to video by sampling frame embeddings and computing cross-modal similarity via parameter-free, sequential, or tight pooling, achieving state-of-the-art results on MSR-VTT, ActivityNet, DiDeMo, and LSMDC (2; 3). Despite these advances, CLIP4Clip’s reliance on full self-attention across T frames incurs $\mathcal{O}(T^2)$ compute and memory costs—rendering it impractical for real-time or low-power deployment (2).

To address the quadratic scaling of attention, linear attention methods such as Performers approximate the softmax kernel via random feature maps (FAVOR+), reducing both time and memory complexity to $\mathcal{O}(T)$ without significant accuracy loss (6). Similarly, sparse 3D attention modules prune redundant patterns in spatio-temporal feature maps, enabling memory-efficient video processing for diffusion and transformer models (7).

Complementing attention optimizations, Spatio-Temporal Auxiliary Networks (STAN) decouple spatial and temporal contextualization into parallel streams before fusion. Liu et al. demonstrate STAN on CLIP features, yielding a +3.7% R@1 boost on MSR-VTT with substantially fewer FLOPs than CLIP4Clip’s full-attention baseline (8). Parallel efforts like Prompt Switch inject learnable spatial–temporal prompt cubes into CLIP’s frozen image encoder, offloading heavy computation to offline preprocessing and securing top performance on MSR-VTT and LSMDC (9).

On the model-compression front, the Lottery Ticket Hypothesis reveals that sparse subnetworks—or “winning tickets”—within dense Transformers can be retrained to match full-model accuracy at high sparsity levels, guiding structured pruning of layers and heads to cut inference cost with minimal performance loss (10).

Rather than processing every frame, adaptive frame-selection strategies such as AdaFrame use lightweight policy networks to observe only 8–9 informative frames per video—matching full-frame accuracy on ActivityNet—dramatically reducing storage and latency (11). Moreover, recent work on frame-feature aggregation shows that replacing mean pooling with excitation and aggregation modules (inspired by max-pooling recalibration) yields +1.2% R@1 improvements on MSR-VTT and similar gains on ActivityNet and DiDeMo (12).

Building on these advances, our project implements a max-pooling similarity calculator, STAN-style modules and multiple frame-selection strategies within the CLIP4Clip framework. We also determine the optimal combination of similarity-calculator head and frame-selection strategy that delivers the best retrieval performance. Our goal is to reduce both training time and inference latency—targeting real-time, low-power video–text retrieval—while maintaining comparable accuracy on standard benchmarks.

2 Our Approach: Fine-tuning CLIP4Clip on Video-Text Dataset

2.1 Problem Description

We aim to explore efficient fine-tuning strategies for the CLIP4Clip model under limited computational resources, targeting the task of video–text retrieval. This task requires the system to retrieve the most relevant video clips from a dataset given a natural language query, or conversely, to identify the most appropriate text descriptions for a given video. It is widely applicable in scenarios such as video search engines, content recommendation, and automated video captioning.

CLIP4Clip is one of the state-of-the-art end-to-end frameworks for video–text retrieval. It jointly models visual features from video frames and semantic features from text queries, achieving strong performance on several benchmark datasets. However, its training process heavily relies on distributed computing with multiple GPUs, making it difficult to reproduce or debug for developers or students with limited hardware resources.

To address this challenge, our goal is to preserve the original structure of CLIP4Clip as much as possible while designing a fine-tuning strategy that supports single-GPU training, enables easy debugging, and remains extensible. Specifically, we propose lightweight improvements in three key areas: optimizing the frame selection strategy, redesigning the similarity calculation module, and

introducing an additional temporal modeling component in the encoder. These components and their performance under varying data scales will be described in the following sections.

2.2 Pipeline Overview

The CLIP4Clip framework leverages pre-trained vision-language models for video-text retrieval through a multi-stage pipeline, as illustrated in Figure 1.

At its core, CLIP4Clip processes video-text pairs through parallel encoding pathways. For video inputs, the pipeline first samples frames according to a predefined strategy (typically uniform sampling). Each frame is processed by CLIP’s Vision Transformer (ViT), generating frame-level embeddings. For textual inputs, the framework tokenizes queries and processes them through CLIP’s text encoder, producing text embeddings.

The similarity calculator then computes cross-modal relevance between video and text representations using one of three approaches: (i) parameter-free mean pooling over frame features, (ii) sequential modeling using transformers or LSTM networks, or (iii) tight cross-modal interaction with transformers. This modular design allows for flexible adaptation of different components while maintaining the core architecture.

Our enhancements focus on optimizing frame selection strategies, introducing additional similarity calculation methods, and incorporating temporal modeling components—all while preserving compatibility with the original framework and ensuring efficient operation on limited computational resources.

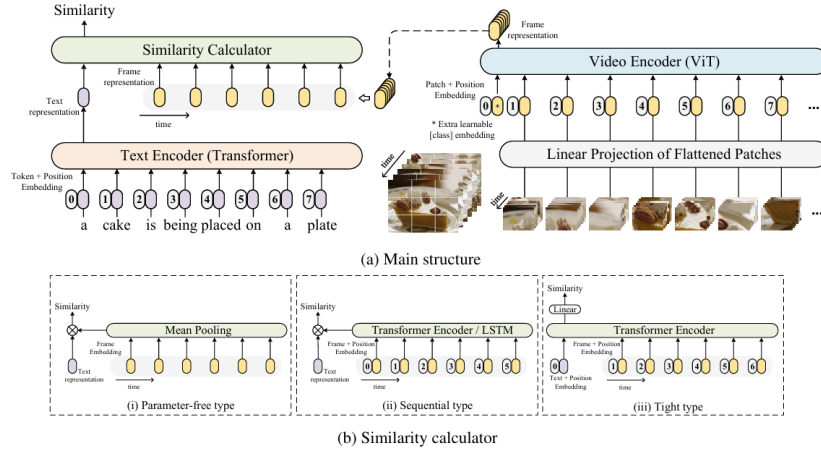


Figure 1: CLIP4Clip pipeline: (a) Main architecture showing parallel video and text encoding paths; (b) Three similarity calculation approaches implemented in the framework.

2.3 Frame Selection Methods

We present a comprehensive analysis of frame selection strategies, beginning with CLIP4Clip’s original uniform sampling as our baseline, followed by our proposed efficient methods. Figure 2 illustrates all approaches. Our implementation maintains a fixed max budget of 8 frames per video, a choice supported by both empirical evidence and computational considerations. This frame limit represents an optimal balance between retrieval performance and computational efficiency, as our experiments show diminishing returns beyond 8 frames while memory and processing requirements increase linearly with frame count. The section below is showing CLIP4Clip’s original frame selection algorithm and how our proposed new algorithm might be more effective. Note: In our algorithms shown below, $n = 8$ for all experiments.

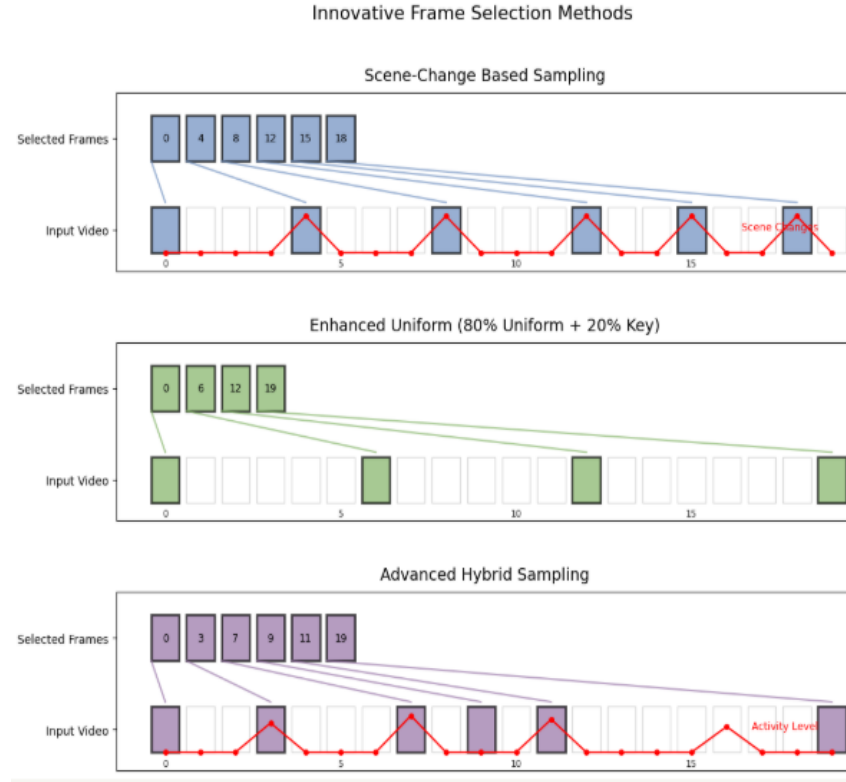


Figure 2: Our proposed new frame method

2.3.1 Frame Selection Algorithms

Algorithm 1 Original Uniform Sampling

Require: Video V , Target frames n
Ensure: Frame sequence F

- 1: $total_frames \leftarrow |V|$
- 2: $step \leftarrow \lfloor total_frames/n \rfloor$
- 3: $F \leftarrow \{\}$
- 4: **for** $i \leftarrow 0$ to $n - 1$ **do**
- 5: $index \leftarrow i \times step$
- 6: **if** $index < total_frames$ **then**
- 7: $F \leftarrow F \cup \{V_{index}\}$
- 8: **end if**
- 9: **end for**
- 10: **return** F

Description: The original approach divides the video into n equal temporal segments and selects one frame from each. This method ensures even coverage across the entire video but cannot adapt to content variations, potentially missing important moments while sampling redundant frames from static scenes.

Algorithm 2 Enhanced Uniform Sampling

Require: Video V , Target frames n , $\alpha = 0.2$
Ensure: Frame sequence F

- 1: $n_{context} \leftarrow \lfloor \alpha \times n \rfloor$
- 2: $n_{uniform} \leftarrow n - n_{context}$
- 3: $F_{context} \leftarrow \{\text{First frame, Last frame}\}$
- 4: $step \leftarrow \lfloor |V|/n_{uniform} \rfloor$
- 5: $F_{uniform} \leftarrow \text{Sample uniformly with } step$
- 6: $F \leftarrow F_{context} \cup F_{uniform}$
- 7: **return** First n frames from F

Description: This method allocates 80% of the frame budget to uniform sampling and 20% to key boundary frames. By prioritizing the beginning and end frames—which often establish context and show outcomes—while maintaining uniform coverage throughout, it preserves both narrative structure and temporal consistency with minimal computational overhead.

Algorithm 3 Scene-Change Based Sampling

Require: Video V , Target frames n , Threshold θ

Ensure: Frame sequence F

```
1:  $D \leftarrow \{\}$  {Frame differences}
2:  $F \leftarrow \{\}$ 
3: for  $i \leftarrow 1$  to  $|V| - 1$  do
4:    $d_i \leftarrow \text{ComputeFrameDifference}(V_i, V_{i+1})$ 
5:   if  $d_i > \theta$  AND  $|F| < n$  then
6:      $F \leftarrow F \cup \{V_i\}$ 
7:   end if
8: end for
9: if  $|F| < n$  then
10:  Fill remaining slots with uniform sampling
11: end if
12: return  $F$ 
```

Description: This content-aware approach analyzes the pixel-wise difference between consecutive frames to identify significant visual transitions. When differences exceed a threshold, frames are selected to capture scene changes, action moments, and important visual cues. Uniform sampling fills any remaining slots, ensuring the method adapts to both high-activity and static videos.

2.3.2 Computational Complexity Analysis

We analyze the computational complexity of our methods in terms of both time and space requirements:

- **Enhanced Uniform:**
 - Time Complexity: $O(n)$ for frame selection, where n is the number of frames
 - Space Complexity: $O(1)$ additional memory beyond baseline
 - Operations: Two additional frame accesses for boundary frames
- **Scene-Change Based:**
 - Time Complexity: $O(N)$ for difference computation, where N is the total video length
 - Space Complexity: $O(N/r)$ where r is the downsampling rate
 - Optimization techniques:
 - * Frame downsampling: Reduces computation by factor r
 - * Difference caching: $O(N/r)$ space for $O(1)$ repeated access
 - * Vectorized operations: Parallel pixel-wise computations
- **Advanced Hybrid:**
 - Time Complexity: $O(N + n \log n)$ for scene detection and sorting
 - Space Complexity: $O(N/r + n)$ for difference storage and frame buffer
 - Optimizations:
 - * Activity score thresholding: Early stopping for efficiency
 - * Shared computations: Reuse of scene detection results
 - * Streaming processing: Constant memory usage

Algorithm 4 Advanced Hybrid Sampling

Require: Video V , Target frames n

Ensure: Frame sequence F

```
1:  $n_{scene} \leftarrow \lfloor n/2 \rfloor$  {Scene-change budget}
2:  $n_{uniform} \leftarrow n - n_{scene}$  {Uniform budget}
3: Compute frame differences across  $V$ 
4:  $F_{scene} \leftarrow \text{Sample } n_{scene} \text{ frames at highest difference points}$ 
5:  $F_{uniform} \leftarrow \text{Sample } n_{uniform} \text{ frames uniformly + boundaries}$ 
6:  $F \leftarrow \text{Combine and deduplicate frames}$ 
7: if  $|F| < n$  then
8:   Add frames using uniform sampling to reach  $n$ 
9: end if
10: return  $F$ 
```

Description: This method allocates half the frames to capture significant visual transitions (detected through frame differences) and half to uniform sampling with boundary frames. By combining these complementary approaches, it provides comprehensive coverage while emphasizing visually important moments, adapting to both static and action-heavy videos.

Table 1: Computational Requirements Comparison

Method	Time Complexity	Space Complexity	GPU Memory
Baseline Uniform	$O(n)$	$O(1)$	1×
Enhanced Uniform	$O(n)$	$O(1)$	1×
Scene-Change	$O(N)$	$O(N/r)$	1.2×
Hybrid	$O(N + n \log n)$	$O(N/r + n)$	1.3×

The additional computational overhead of our methods is justified by improved frame selection quality and is negligible compared to the cost of CLIP feature extraction and cross-modal matching operations.

2.4 Video Embedding Generation

Algorithm 5 generates video-level embeddings by uniformly sampling n frames from a video, preprocessing them using CLIP’s processor (resizing, normalization), and extracting frame-wise visual features via CLIP’s image encoder. Each frame is passed through CLIP’s image encoder (originally designed for images) to generate an embedding. These frame embeddings are aggregated via temporal mean pooling and L2-normalized for cosine similarity comparisons.

The algorithm ensures robustness by returning null if frame extraction fails. It is designed for video-text retrieval, leveraging CLIP’s cross-modal capabilities to align video embeddings with textual representations. This method assumes uniform frame importance and ignores temporal dynamics.

Algorithm 5 Video Embedding Generation

Require: CLIP model M , CLIP processor P , Video path $path$, Frames to sample n
Ensure: Video-level embedding normalized to unit length

- 1: $frames \leftarrow \text{ExtractFrames}(path, n)$ {Call Algorithm 2}
- 2: **if** $frames$ is empty **then**
- 3: **return** null
- 4: **end if**
- 5: $inputs \leftarrow P(frames)$ {Process frames with CLIP processor}
- 6: $frame_embeddings \leftarrow M.\text{ImageEncoder}(inputs)$ {Each frame passes through CLIP’s image encoder}
- 7: $video_embedding \leftarrow \text{Mean}(frame_embeddings)$ {Temporal aggregation}
- 8: $video_embedding \leftarrow \frac{video_embedding}{||video_embedding||}$ {L2-normalization}
- 9: **return** $video_embedding$

2.5 Text Embedding Generation

The text embedding process converts textual queries into vector representations aligned with CLIP’s multimodal embedding space, enabling cross-modal retrieval. Each caption or text query is processed by CLIP’s text encoder to generate a text embedding. Raw text is first tokenized and preprocessed by CLIP’s text processor, standardizing input (truncation, padding, special tokens) and converting it into tensor format. These processed inputs are then passed through CLIP’s pretrained text encoder to extract high-dimensional features.

The resulting embeddings are L2-normalized to unit length, ensuring compatibility with cosine similarity metrics. This normalization allows direct comparison between text and video embeddings in a shared semantic space. By mirroring the video embedding pipeline, this method maintains consistency in representation learning, improving video-text retrieval accuracy.

2.6 Similarity Calculation

In the original CLIP4Clip framework, a sequence of frame embeddings is aggregated into a single video representation using one of four similarity calculation strategies: *mean pooling*, *sequential LSTM*, *sequential transformer*, and *tight transformer*.

While these methods offer varying degrees of temporal modeling, they tend to dilute the contribution of high-salience frames by averaging or distributing attention uniformly. To better capture key visual moments in a video, we introduce a lightweight alternative, *max pooling*, which selects the maximum activation across frames in each embedding dimension. This strategy enhances the influence of the most informative frames while avoiding unnecessary smoothing.

The following section provides a detailed description of all five similarity calculators.

Mean Pooling:

$$v = \frac{1}{m} \sum_{t=1}^m F_t, \quad (1)$$

where $F_t \in \mathbb{R}^d$ is the embedding of frame t , and m is the total number of frames.

SeqLSTM: A bidirectional LSTM processes $\{F_1, \dots, F_m\}$ and uses the final hidden state as the video embedding v .

SeqTransformer: A standard Transformer encoder ingests the frame sequence $\{F_t\}$; the output tokens are either mean-pooled or a special “[CLS]” token is taken as v .

Tight Transformer: Cross-modal attention is applied between text and video at each Transformer layer; the final pooled output serves as v .

Max Pooling (new):

$$v[d] = \max_{1 \leq t \leq m} F_t[d] \quad (\forall d = 1, \dots, d), \quad (2)$$

i.e. the maximum activation across all frames in each dimension.

2.7 Similarity Retrieval

Let $T \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{M \times d}$ be the matrices of text and video embeddings, respectively, where each row is ℓ_2 -normalized. We compute the similarity matrix

$$S = TV^\top \in \mathbb{R}^{n \times M}, \quad (3)$$

so that

$$s_{ij} = T_i \cdot V_j = \cos(t_i, v_j). \quad (4)$$

For each text query i , we retrieve the top- k videos by

$$(\text{scores}, \text{indices}) = \text{torch.topk}(S[i], \min(k, M)). \quad (5)$$

Mapping each index back to its video filename yields an ordered list

$$(\text{filename}_j, s_{ij}) \quad (j = 1, \dots, k), \quad (6)$$

which constitutes the k most relevant videos for query i .

2.8 STAN

What Does STAN Add to CLIP4Clip?

In the original CLIP4Clip architecture, video features are extracted by sampling multiple frames and encoding them using the pre-trained CLIP image encoder. These frame-level features are then aggregated—typically using mean pooling or attention—to form a fixed-length video representation, which is compared with the text embedding for retrieval(2).

However, CLIP4Clip has limited capacity to model fine-grained temporal dynamics, since CLIP itself is an image-level encoder and lacks built-in temporal awareness. This is where the STAN (Spatial-Temporal Auxiliary Network) module becomes beneficial.

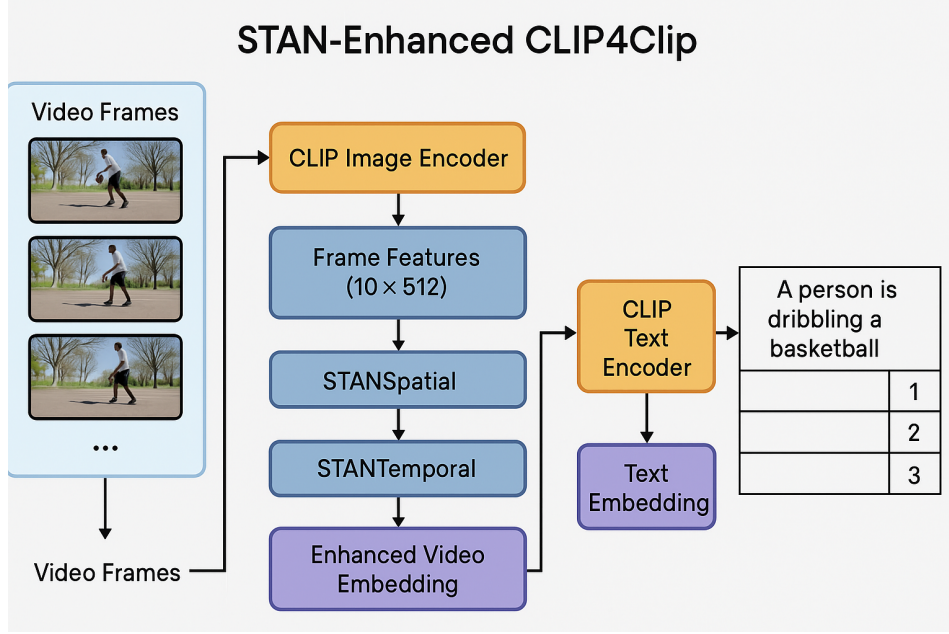


Figure 3: STAN-Enhanced CLIP4Clip Pipeline: Frame features extracted by the CLIP image encoder are refined using STAN’s spatial and temporal attention modules to produce a rich video embedding, which is matched against text embeddings for retrieval.

Role of STAN in CLIP4Clip

STAN augments the CLIP4Clip pipeline with explicit spatial-temporal modeling capabilities by:

- **Decomposing Temporal Modeling:** STAN separates the modeling of spatial features (individual frame content) and temporal relationships (across frames), allowing each aspect to be optimized more effectively.
- **Branch-Based Feature Fusion:** STAN introduces a fusion mechanism that blends global video context (obtained via mean pooling) with localized, frame-specific information (e.g., from the first frame), controlled by a tunable weighting parameter α :

$$\text{fused_output} = \alpha \cdot \text{visual_output} + (1 - \alpha) \cdot x$$

where visual_output denotes the average-pooled feature vector over all frames, and x is a key frame feature. This mechanism allows the model to emphasize either holistic scene understanding (large α) or fine-grained, moment-specific cues (small α), which is crucial in video where temporal details matter.

- **Improving Contextualization:** By producing a temporally-aware and frame-sensitive representation, STAN enhances CLIP4Clip’s ability to align video content with natural language queries more precisely.

2.9 Evaluation Methodology

Key evaluation metrics for video-text pairing include Recall@K, Median Rank and Mean Rank, as summarized in Table 2.

In our evaluation on the MSRVT dataset, we implement standard metrics for video-text retrieval by first calculating retrieval metrics (R@1, R@5, R@10) through ranking each video based on how many other videos have higher similarity scores, then measuring recall@K to quantify how many correct matches appear in the top-K predictions.

Metric	Description
R@1	Percentage of queries for which the correct video is the top-ranked result.
R@5 and R@10	Percentage of queries for which the correct video appears in the top-5 or top-10 results.
Median Rank	The median position of the correct video (or text) across all queries, showing the middle value of retrieval ranks.
Mean Rank	The average rank position of the correct video (or text) across all queries, indicating overall retrieval quality.

Table 2: Evaluation Metrics for Video-Text Retrieval

3 Experiment

3.1 Dataset

We conduct our experiments on MSR-VTT dataset (14), one of the most widely used benchmarks for video-text retrieval. It contains 10,000 video clips collected from a commercial video website, with durations ranging from 10 to 32 seconds. Each video is annotated with approximately 20 natural language sentences, resulting in a total of around 200,000 video-text pairs.

Following the protocol of prior works such as CLIP4Clip (2), X-CLIP (3) and STAN (4), we use the official split with 6,513 videos for training, 497 for validation, and 2,990 for testing. Each caption can be used to retrieve the corresponding video (text-to-video retrieval), and vice versa (video-to-text retrieval). This dataset enables the evaluation of both modalities and is suitable for contrastive video-text embedding models.

3.2 Experimental Setup

To adapt the CLIP4Clip pipeline for training on a single GPU, we removed all distributed training components and simplified the codebase for ease of debugging and local runs.

We adopt the same hyperparameters as the CLIP4Clip paper for most settings: learning rate = 1e-4, text length = 32, and a batch size of 64 (the maximum supported on A100/V100). We freeze the first 6 layers of the image encoder, following findings that this accelerates training while preserving performance. To determine the number of frames per video, we tested 8–12 and found that 8 performs best, likely due to reduced visual redundancy. We also adopt the loss function, optimizer, and learning rate scheduler that were reported to yield the best performance in the original CLIP4Clip paper. Specifically, we use the Symmetric InfoNCE loss, Adam optimizer, a learning rate decay of 0.9, and enable mixed precision training (FP16) for improved efficiency.

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^N \left[\log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ij}/\tau}} + \log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ji}/\tau}} \right]$$

Symmetric InfoNCE loss

To verify that this configuration can scale to the full MSR-VTT dataset under single-GPU conditions, we replicated the original CLIP4Clip baseline on the full set using our setup. We observed smooth convergence and competitive retrieval performance. Detailed training loss and performance trends are shown in Table 3.

Epoch	Train Loss	Text-to-Video			Video-to-Text		
		R@1	R@5	R@10	R@1	R@5	R@10
1	0.126967	38.3	62.3	73.3	38.4	64.7	76.1
2	0.092233	38.6	63.4	75.9	39.8	65.8	77.0
3	0.088145	39.1	63.7	76.3	40.2	65.8	77.5
4	0.085432	39.5	64.0	76.8	40.5	66.0	78.0
5	0.083210	39.8	64.5	77.2	40.8	66.2	78.3

Table 3: Training Loss and Retrieval Metrics per Epoch on Full MSR-VTT Dataset

Across both experimental tracks, we use the same configuration described above and adopt uniform frame sampling and mean pooling as our shared baseline, which is adapted from CLIP4Clip for improved training efficiency on limited hardware. To accelerate the exploration of module combinations, most experiments are conducted on a 0.1 subset of MSR-VTT (999 videos).

3.3 Experiment Design

Our experiments are structured into two tracks to investigate: (1) The effect of frame sampling methods and simulator modules (two-stage evaluation); (2) The impact of encoder-level modifications using STAN.

Track 1: The effect of frame sampling methods and simulator modules

Preliminary Testing We use a 0.1 subset of MSR-VTT (999 videos) to evaluate modules under controlled conditions.

- With *uniform* frame sampling fixed, we compare five similarity calculators: *mean pooling*, *max pooling*, *sequential LSTM*, *sequential transformer*, and *tight transformer*.
- With *mean pooling* calculator fixed, we evaluate frame sampling strategies: *hybrid allocation*, *scene-changed sampling*, and *80% uniform + 20% context*.

All models are trained for 5 epochs using a batch size of 64 (maximum on A100/V100), a learning rate of $1e-4$, and a max text length of 32, following CLIP4Clip defaults. The image encoder freezes the first 6 layers, which improves training speed without harming performance. We also tested 8–12 frames per video and found 8 frames performs best due to reduced visual noise.

Combination Evaluation Based on Stage 1 results, we select the top 3 similarity calculators and top 4 frame sampling methods. All 12 combinations are trained for 20 epochs using the same configuration.

Track 2: The impact of encoder-level modifications using STAN

For encoder-level changes such as integrating STAN blocks, we train directly on the full MSR-VTT dataset without prior module testing. Hyperparameters are consistent with CLIP4Clip, except for a slightly smaller batch size due to GPU memory constraints.

3.4 Preliminary Results (Track 1)

We report retrieval performance using Recall@K (R@1, R@5, R@10), Median Rank (MedR), and Mean Rank (MeanR) for both text-to-video and video-to-text tasks, based on 5-epoch training on the MSR-VTT 0.1 subset.

Similarity Calculators Among the five evaluated modules, *max pooling* achieves the best Mean Rank while maintaining competitive recall scores, indicating its strength in capturing informative frames. *Sequential LSTM* and *sequential transformer* slightly outperform *mean pooling* in R@1 and R@5, while *tight transformer* underperforms across all metrics. Overall performance follows the trend:

$$\text{Mean Pooling} \approx \text{Sequential LSTM} \approx \text{Sequential Transformer} \\ > \text{Max Pooling} \gg \text{Tight Transformer}$$

Frame Sampling Methods The standard *uniform sampling* remains a strong baseline. However, newly proposed strategies such as *hybrid allocation + sampling*, *scene-changed sampling*, and *80% uniform + 20% key context* yield comparable or better performance on several metrics. In contrast, *tail-only* and *head-only* methods in original CLIP4Clip perform significantly worse. The overall effectiveness can be summarized as:

$$\text{Scene-changed} \approx \text{Uniform} \approx \text{80\%Uniform + 20\%KeyContext} \\ \approx \text{Hybrid Allocation} > \text{Head-only} > \text{Tail-only}$$

Detailed per-module evaluation results are available in Appendix C. These findings demonstrate that several proposed modules outperform or rival the baseline configuration (uniform + mean pooling) in both retrieval accuracy and ranking quality. However, while our alternative frame sampling methods

show comparable effectiveness to the baseline *uniform* strategy, they provide greater flexibility and diversity in capturing informative frames.

So, we select the top 3 similarity calculators and 4 frame sampling methods for combined evaluation in next combination evaluation section.

3.5 Preliminary Result (Track 2)

To evaluate the effectiveness of the STAN module and the impact of different α values, we conducted preliminary tests on a subset of 50 videos from the MSR-VTT dataset. We assessed retrieval performance using standard metrics: Recall@1 (R@1), Recall@5 (R@5), Recall@10 (R@10), and Median Rank. The results indicate that an α value of 0.5 provides the best balance between global and local features, leading to improved retrieval performance.

α	R@1	R@5	R@10	Median Rank
0.0	0.0400	0.1400	0.2800	20.0
0.3	0.0200	0.1400	0.2600	21.0
0.5	0.0400	0.1200	0.2600	24.0
0.7	0.0200	0.1400	0.2000	23.0
1.0	0.0200	0.0800	0.2000	29.0

Table 4: Retrieval performance metrics for different alpha values in the STAN module.

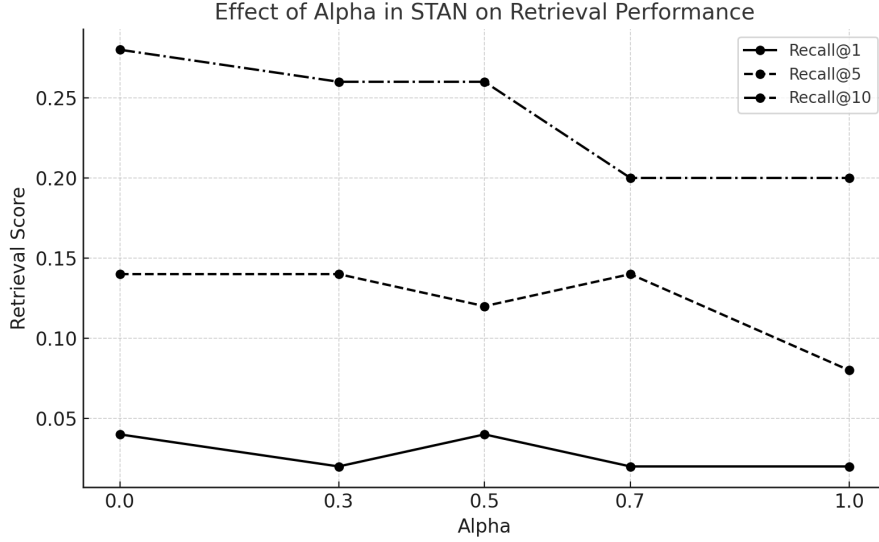


Figure 4: Retrieval performance (R@1, R@5, R@10) under different α values in the STAN module.

Interpretation:

- $\alpha = 0.0$: Utilizing only the first frame features yields the highest Recall@10 (0.2800) and the lowest median rank (20.0), indicating that initial frame information is particularly salient in this setup.
- $\alpha = 0.3, \alpha = 0.5$: Introducing a blend of first-frame and average-pooled features leads to a slight decline in performance, with median ranks increasing to 21.0 and 24.0, respectively. This suggests that the fusion of features may introduce redundancy or noise in this context.
- $\alpha = 0.7, \alpha = 1.0$: As the model increasingly emphasizes average-pooled features, performance further deteriorates, with Recall@1 dropping to 0.0200 and median rank rising to 29.0 at $\alpha = 1.0$. This trend implies that over-reliance on averaged features may dilute critical temporal information captured in specific frames.

3.6 Combined Results

To further understand the joint effects of frame selection and simulator modules, we conducted a full combination evaluation using 12 variants (3 simulators $\times$ 4 frame selection strategies), trained for 20 epochs on the MSR-VTT 0.1 subset. The results are summarized in Figure 5.

Framing and simulator combination	Text-to-Video					Video-to-Text				
	R@1	R@5	R@10	Median R	Mean R	R@1	R@5	R@10	Median R	Mean R
MeanP+Uniform	39.5	63.6	74.6	3.0	22.3	35.9	63.3	73.9	3.0	18.3
MeanP+Scene-change	38.3	62.8	74.2	3.0	22.3	34.8	61.1	72.6	3.0	19.2
MeanP+80%Uniform+20%Key	37.7	63.4	73.6	3.0	22.3	35.2	60.6	71.3	3.0	19.1
MeanP+Hybrid	37.2	62.9	74.3	3.0	20.9	35.6	65.0	76.1	3.0	17.7
SeqLSTM+Uniform	39.6	64.8	74.6	2.0	21.1	37.0	64.4	75.5	3.0	17.4
SeqLSTM+Scene-change	38.7	64.2	75.2	3.0	21.2	37.4	64.1	75.2	3.0	17.2
SeqLSTM+80%Uniform+20%Key	37.9	63.8	74.2	3.0	21.0	35.4	62.1	72.2	3.0	19.2
SeqLSTM+Hybrid	37.8	63.5	74.5	3.0	21.4	34.9	63.9	75.1	3.0	17.6
SeqTransf+Uniform	39.6	64.4	75.4	3.0	22.2	35.5	62.1	73.8	3.0	19.3
SeqTransf+Scene-change	37.7	63.5	75.2	3.0	22.3	34.4	59.0	71.8	3.0	21.0
SeqTransf+80%Uniform+20%Key	37.5	62.6	74.3	3.0	24.1	31.9	60.5	71.5	3.0	23.1
SeqTransf+Hybrid	38.2	63.8	74.6	3.0	22.7	34.5	62.3	71.8	3.0	21.4

Figure 5: Performance of framing and simulator combinations on the MSR-VTT 0.1 subset (Epoch 5). Each row represents a pairing of one simulator with a frame selection strategy. Metrics include Recall@K, Median Rank, and Mean Rank for both text-to-video and video-to-text retrieval.

While uniform frame sampling is the default in CLIP4Clip, our proposed alternatives—scene-change based sampling, 80% uniform + 20% key context, and hybrid allocation—achieve similarly competitive performance across text-to-video and video-to-text retrieval tasks. Specifically, combinations like MeanP+Uniform, SeqLSTM+Uniform, and SeqTransf+Uniform each reach R@1 scores of 39.5–39.6 in the T2V setting, matching the strongest baseline configurations.

In addition, for simulator types such as MeanP, pairing with context-aware or adaptive sampling strategies (e.g., 80% + 20% key frames) maintains robust retrieval accuracy and consistent median rank, while sometimes yielding improvements in mean rank. This indicates that these frame selection methods introduce little visual noise while preserving semantic quality.

These results suggest that our lightweight module and frame-level enhancements offer a flexible and modular alternative to the original CLIP4Clip baseline, enabling easy experimentation while maintaining retrieval performance.

In addition, since we extend the training of these configurations to 20 epochs to investigate long-term trends. As illustrated in Appendix D, we observe continued improvement over epochs, supporting the robustness and optimization potential of our method combinations.

Result of integrating Spatial-Temporal Auxiliary Network in CLIP4Clip framework vary by fusion hyperparameter, α . With our preliminary test, half of fusion framed by STAN and half framed by ViT-B/32 generates an optimal recall value. The modified architecture has been tested on a subset of 50 video samples using CPU resources, validating the structural soundness and functional correctness of the updated pipeline. Full-scale training and evaluation on larger datasets are forthcoming.

4 Conclusion

Our work demonstrates the feasibility and effectiveness of fine-tuning the CLIP4Clip framework under constrained computational resources (1 GPU) while preserving retrieval accuracy. Through systematic exploration of modular enhancements—including frame selection strategies, similarity calculation methods, and the integration of spatial-temporal auxiliary networks (STAN)—we show that lightweight architectural modifications can yield performance comparable to the original configuration on standard benchmarks such as MSR-VTT.

Empirical evaluations reveal that alternative frame sampling methods, such as scene-change-based and hybrid allocation, provide improved temporal diversity with minimal computational overhead.

Similarly, max pooling emerges as a promising similarity calculator, offering strong ranking performance while avoiding the complexity of sequential modeling. The integration of STAN further augments temporal representation learning, with results indicating that local frame-level features often contribute more significantly to retrieval performance than global averaged representations.

Overall, our work provides a scalable and extensible approach to efficient video-text retrieval, enabling broader applicability of CLIP-based architectures for videos in real-time and low-computation settings.

5 Future Directions

Our project could be extended in the following directions:

- **Dataset Expansion:** Incorporating larger and more diverse video datasets to improve the generalizability and robustness of experimental results.
- **Multimodal Enhancement:** Integrating language models to enable higher-level tasks such as video summarization or question answering.
- **Domain-Specific Training:** Fine-tuning on domain-specific video corpora (e.g., astronomy, sports, education) to better support retrieval tasks within specialized contexts.
- **Temporal Modeling Enhancement:** Incorporating positional encodings and more sophisticated temporal modeling techniques to better capture sequential information in videos. This would address the current limitation of treating frames as relatively independent units and potentially improve performance on action-oriented queries where temporal order is crucial.
- **Video Typology Analysis:** Developing a systematic framework for categorizing videos based on their content characteristics (e.g., static vs. dynamic, narrative vs. instructional) and analyzing how different sampling strategies perform across these categories. This would enable more targeted optimization of frame selection methods for specific video types.

6 Team Roles and Repository

6.1 Team Roles

- **Xiaoyang Ma:** Responsible for researching all new frame selection methods and conducting the corresponding training and testing.
- **Sophia Wang:** Conducted literature review, full-set training on 1 GPU, algorithm implementation, and code integration of CLIP4Clip + STAN module.
- **Ruan Hang:** Integrated new similarity calculators (max pooling) and handled associated experiments to finetune and optimize the model.
- **Sijia Ma:** Set up the project environment and pipeline, and contributed to training and testing procedures.

6.2 GitHub Repository

All code and documentation are available in our public repository:

https://github.com/sijiasiga/CLIP_video

References

- [1] Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning Transferable Visual Models From Natural Language Supervision. *arXiv preprint arXiv:2103.00020*.
- [2] Luo, H., Ji, L., Zhong, M., Chen, Y., Lei, W., Duan, N., & Li, T. (2021). CLIP4Clip: An Empirical Study of CLIP for End-to-End Video Clip Retrieval. *arXiv preprint arXiv:2104.08860*.
- [3] Ma, Y., Xu, G., Sun, X., Yan, M., Zhang, J., & Ji, R. (2022). X-CLIP: End-to-End Multi-grained Contrastive Learning for Video-Text Retrieval. *arXiv preprint arXiv:2207.07285*.
- [4] Liu, R., Huang, J., Li, G., Feng, J., Wu, X., & Li, T. H. (2023). Revisiting Temporal Modeling for CLIP-based Image-to-Video Knowledge Transferring. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- [5] Liu, R., Huang, J., Gao, W., Li, T. H., & Li, G. (2023). Mug-STAN: Adapting Image-Language Pretrained Models for General Video Understanding. *arXiv preprint arXiv:2311.15075*.
- [6] Choromanski, K., Likhoshesterov, V., Dohan, D., et al. (2020). Rethinking Attention with Performers. *arXiv preprint arXiv:2009.14794*.
- [7] Tian, K., Li, S., & Huang, Y. (2023). Sparse 3D Attention for Memory-Efficient Video Processing. *arXiv preprint arXiv:2311.15769*.
- [8] Liu, R., Huang, J., Li, G., Feng, J., Wu, X., & Li, T. H. (2023). Revisiting Temporal Modeling for CLIP-Based Image-to-Video Knowledge Transferring (STAN). *arXiv preprint arXiv:2301.11116*.
- [9] Deng, C., Chen, Q., Qin, P., Chen, D., & Wu, Q. (2023). Prompt Switch: Efficient CLIP Adaptation for Text-Video Retrieval. *arXiv preprint arXiv:2308.07648*.
- [10] Chen, T., Goodfellow, I., & Shlens, J. (2020). The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [11] Wu, Z., Xiong, C., Ma, C.-Y., Socher, R., & Davis, L. S. (2019). AdaFrame: Adaptive Frame Selection for Fast Video Recognition. *arXiv preprint arXiv:1811.12432*.
- [12] Jing, X., Yang, G., & Chu, J. (2024). An Empirical Study of Excitation and Aggregation Design Adaptions in CLIP4Clip for Video-Text Retrieval. *arXiv preprint arXiv:2406.01604*.
- [13] Rasheed, M., Wu, Q., & Xie, S. (2023). Fine-Tuned CLIP Models Are Efficient Video Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [14] Xu, J., Mei, T., Yao, T., & Zhang, Y. (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 (CVPR)* (pp. 5288–5296).

Appendix

A CLIP4Clip Model Details

Module	Input Shape	Output Shape	Notes
Vision Encoder (ViT-B/32)			
Frame Input	(128, 3, 224, 224)	–	16 videos $\times$ 8 frames = 128 frames
Patch Embedding	(128, 3, 224, 224)	(128, 49, 768)	32 $\times$ 32 patches $\rightarrow$ 7 $\times$ 7 = 49
+ Positional Encoding	(128, 49, 768)	(128, 50, 768)	Add [CLS] token
Transformer Layers ($\times$ 12)	(128, 50, 768)	(128, 50, 768)	12-layer ViT-B
CLS Token Extraction	(128, 50, 768)	(128, 768)	Use CLS token output
Linear Projection	(128, 768)	(128, 512)	Project to shared embed space
Reshape to video	(128, 512)	(16, 8, 512)	Group 8 frames per video
Text Encoder (CLIP Text Transformer)			
Tokenized Input	(16, 32)	–	Text token indices
Embedding + Pos Encoding	(16, 32)	(16, 32, 512)	Word embedding + position
Transformer Layers ($\times$ 12)	(16, 32, 512)	(16, 32, 512)	12-layer Transformer
Final Embedding	(16, 32, 512)	(16, 512)	Use last token or mean pool
Similarity Module (meanP)			
Mean Pooling (Video)	(16, 8, 512)	(16, 512)	Temporal mean across frames
Cosine Similarity Matrix	(16, 512) $\times$ (16, 512)	(16, 16)	Similarity score matrix
Cross-Encoder (Optional: cross-base)			
Concat Text + Video	(16, 40, 512)	(16, 40, 512)	32 tokens + 8 frames
Transformer Layers ($\times$ 4)	(16, 40, 512)	(16, 40, 512)	4-layer encoder
CLS Output	(16, 40, 512)	(16, 512)	Output pooled embedding

Table 5: CLIP4Clip Layer-wise Input and Output Shapes

B STAN Layer Details

Module	Input Shape	Output Shape	Description
STAN Layer			
Input Embeddings	(128, 8, 49, 512)	–	128 videos $\times$ 8 frames $\times$ 49 patches $\times$ 512-dim features
Reshape	(128, 8, 49, 512)	(128, 392, 512)	Flatten frames and patches: 8 $\times$ 49 = 392
Intra-Frame Attention	(128, 392, 512)	(128, 392, 512)	Applies self-attention within each frame’s patches
Cross-Frame Attention	(128, 392, 512)	(128, 392, 512)	Models temporal dependencies across frames
Feed-Forward Network	(128, 392, 512)	(128, 392, 512)	Applies non-linear transformations
Layer Normalization	(128, 392, 512)	(128, 392, 512)	Stabilizes training and facilitates gradient flow
Reshape Back	(128, 392, 512)	(128, 8, 49, 512)	Reshape to original frame and patch dimensions

Table 6: STAN Layer-wise Input and Output Shapes (B=128, T=8)

C Additional Experimental Results (5 epochs)

Text-to-Video and Video-to-Text Performance Metrics (Epoch 5)

Simulator Type	Text-to-Video					Video-to-Text				
	R@1	R@5	R@10	Median R	Mean R	R@1	R@5	R@10	Median R	Mean R
Max Pooling	35.8	61.5	72.5	3.0	23.3	33.5	60.7	72.0	3.0	19.1
Mean Pooling	36.8	64.0	75.4	3.0	21.4	36.9	63.8	73.9	3.0	17.2
Seq LSTM	38.6	63.5	73.8	3.0	21.9	36.6	63.2	74.6	3.0	18.0
Seq Transf	38.4	64.3	74.8	3.0	20.9	35.0	60.4	73.9	3.0	18.9
Tight Transf	26.7	56.5	68.9	4.0	30.3	26.7	55.9	66.7	4.0	33.4

Figure 6: Performance of different simulator modules (5 epochs).

Text-to-Video and Video-to-Text Performance Metrics (Epoch 5)

Frame Selection Type	Text-to-Video					Video-to-Text				
	R@1	R@5	R@10	Median R	Mean R	R@1	R@5	R@10	Median R	Mean R
#2. Uniform(Original Best)	36.9	64.3	75.3	3.0	21.4	36.7	63.8	74.0	3.0	17.2
#8. 80%Uniform+20%Key Context	36.7	64.1	74.9	3.0	21.1	35.7	62.9	74.8	3.0	17.1
#9. Hybrid Allocation + Sampling	36.2	62.6	75.0	3.0	22.2	35.2	62.9	74.7	3.0	18.3
#5. Scene-changed Sampling	36.7	63.4	74.8	3.0	21.8	36.0	63.8	73.4	3.0	17.4
#0. Head Frame Selection(Original)	34.7	61.6	73.6	3.0	24.6	34.1	63.0	73.4	3.0	19.2
#1. Tail Frame Selection(Original)	33.0	57.7	69.9	4.0	29.4	31.4	57.8	69.4	4.0	25.8

Figure 7: Performance of different frame sampling methods (5 epochs).

D Additional Experimental Results (20 epochs)

This appendix provides extended training performance for each simulator type across 20 epochs. The following plots illustrate retrieval accuracy improvements over time:

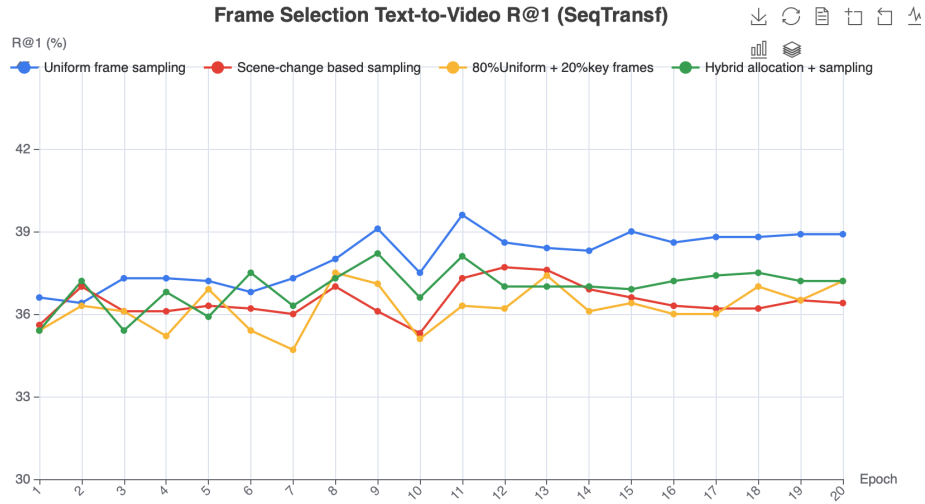


Figure 8: SeqTransformer Text-to-Video R@1 across 20 epochs

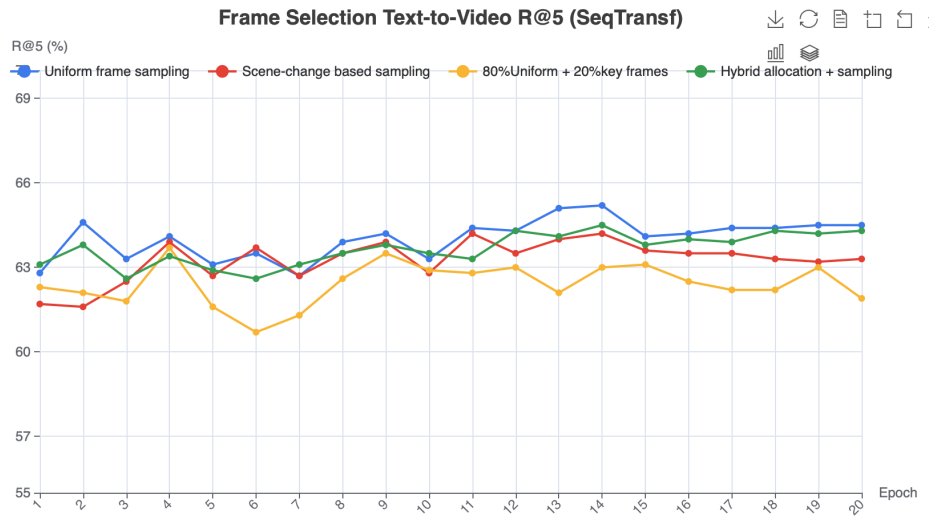


Figure 9: SeqTransformer Text-to-Video R@5 across 20 epochs

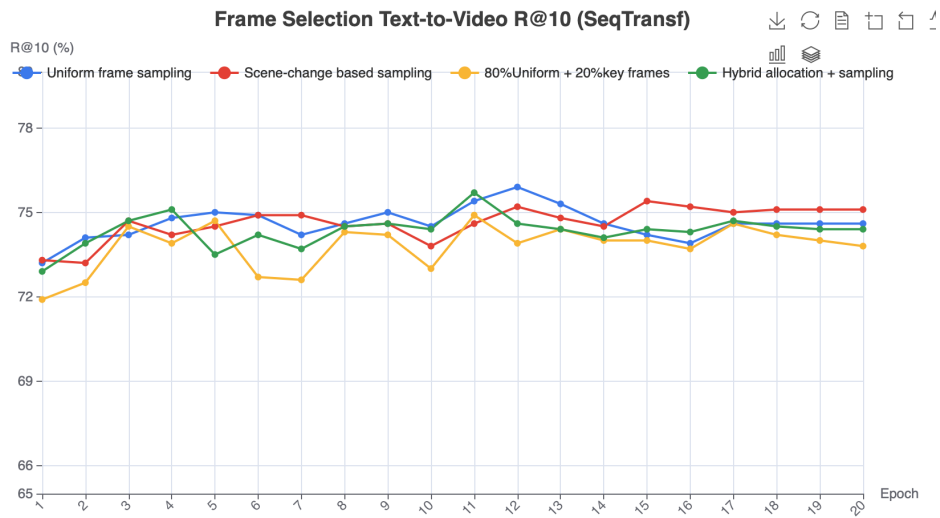


Figure 10: SeqTransformer Text-to-Video R@10 across 20 epochs

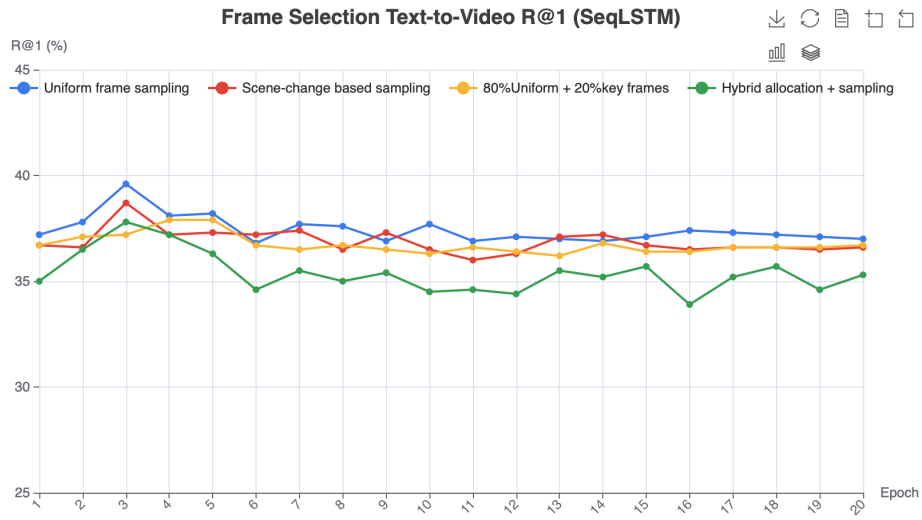


Figure 11: SeqLSTM Text-to-Video R@1 across 20 epochs

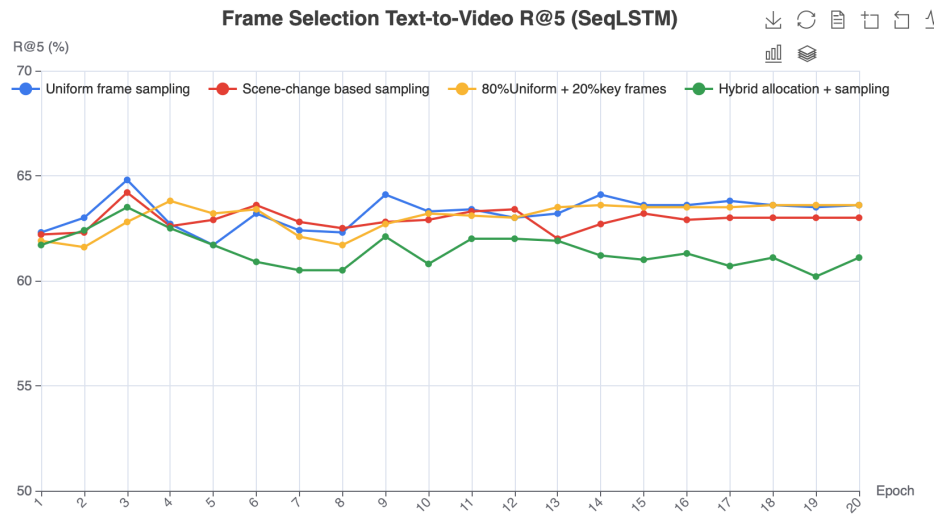


Figure 12: SeqLSTM Text-to-Video R@5 across 20 epochs

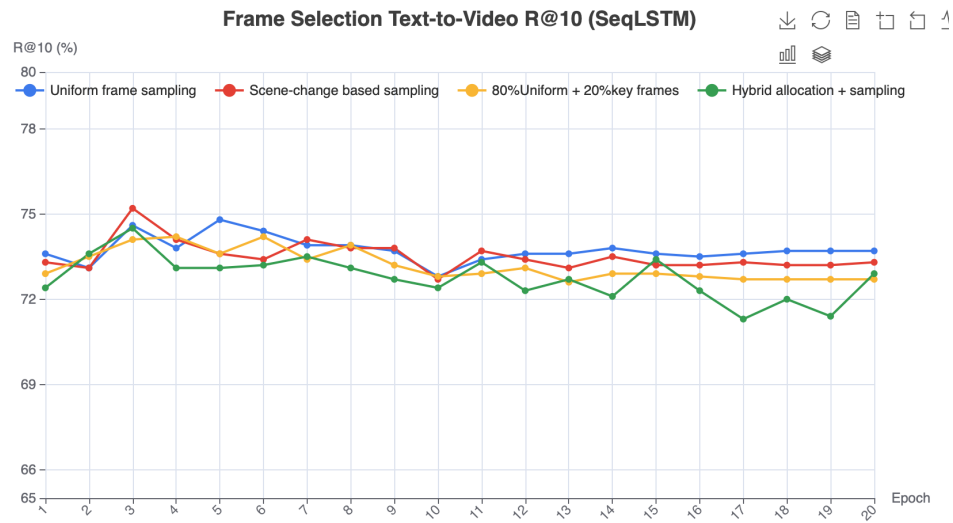


Figure 13: SeqLSTM Text-to-Video R@10 across 20 epochs

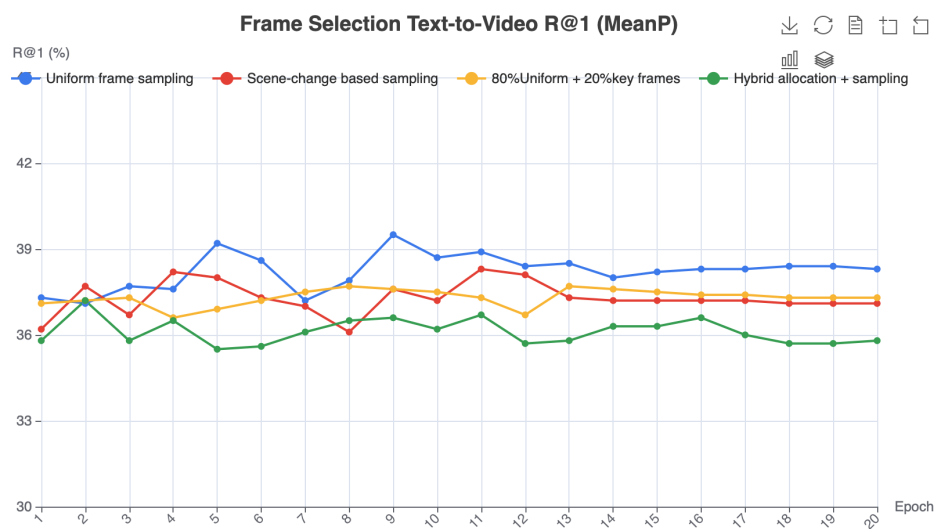


Figure 14: Mean Pooling Text-to-Video R@1 across 20 epochs

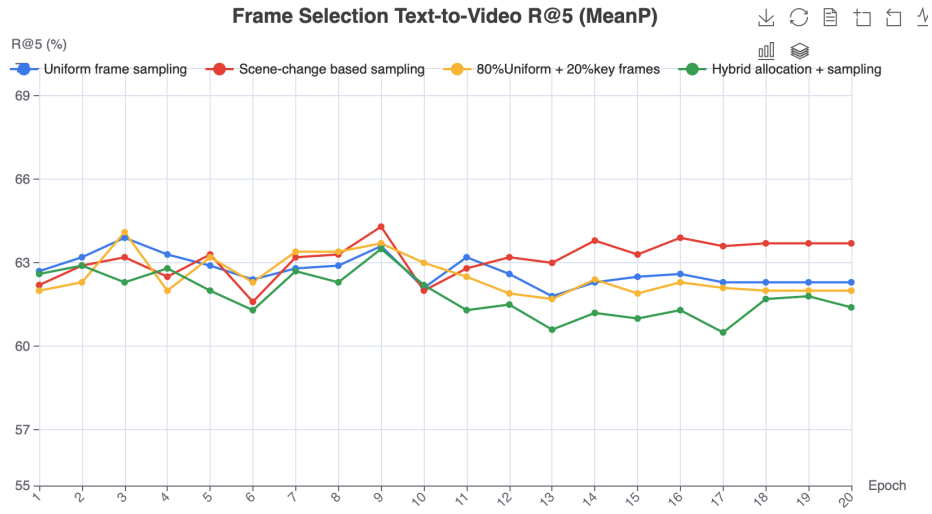


Figure 15: Mean Pooling Text-to-Video R@5 across 20 epochs

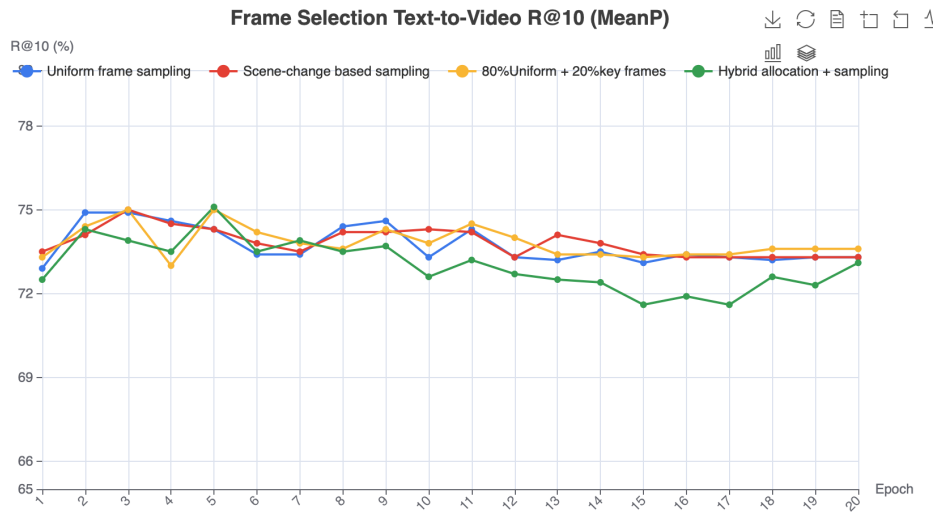


Figure 16: Mean Pooling Text-to-Video R@10 across 20 epochs