MMSAMPLER: EFFICIENT FRAME SAMPLER FOR MULTIMODAL VIDEO RETRIEVAL

Zhiming Hu [*1] **Angela Ning Ye** [*1] **Iqbal Mohomed** [1]

ABSTRACT

We study the problem of natural language-based video retrieval, the task of finding relevant videos given natural language search queries. Most recent state-of-the-art (SOTA) approaches would embed the video and query separately and map the video and query embeddings into a joint latent space to calculate a similarity score between them. To learn a video representation, existing solutions generally use all the frames or sample a subset of frames from the video using uniform sampling. The former solution could be computationally prohibitive while the latter may inject noise from uninformative frames into the final video representation. To this end, we propose mmSampler, a learning-based sampler, to adaptively select *salient frames* to represent the videos for **m**ulti**m**odal video retrieval. mmSampler can greatly reduce the computational overhead for video representation without affecting the retrieval performance. We learn a lightweight policy network to decide whether to further process or discard a frame. By adopting the Gumbel-Softmax trick, we train the sampler jointly with the video retrieval model end-to-end in an efficient manner. Experimental results on benchmark datasets such as ActivityNet, DiDeMo and MSRVTT demonstrate that mmSampler achieves improved retrieval performance while saving as much as 43% GFLOPs per video.

1 INTRODUCTION

Joint understanding of vision and language is one of the key research topics in the machine learning community. One task under this umbrella is natural language-to-video retrieval, which returns relevant videos from a pool of candidate videos given a textual query. The rapid rise of user-generated videos on mobile devices (from flagship to feature phones) as a global phenomenon has made efficient retrieval of videos an important practical problem.

The general approach for video and natural language understanding involves learning vector representations for the videos and the queries in a joint embedding space. In the learned space, semantically correlated content will be closer in distance (*e.g.*, cosine similarity), as depicted in Fig. 1. A key question that our work addresses is how to embed the visual frames efficiently into a video representation.

Existing work on video retrieval handles the embedding of visual frames in two main approaches. The first line of work extracts visual features for all the frames in a video and aggregates the frame features into a compact video representation (Miech et al., 2018; Liu et al., 2019). Such approaches seek to optimize the retrieval performance while ignoring the computational overhead associated with extracting all the frame features. Another line of work adopts uniform sampling by either sampling a fixed number of frames per video (Chen et al., 2018; Bain et al., 2021; Luo et al., 2021) or sampling at a fixed frame rate (Yu et al., 2018) to select a subset of frames to represent a video. These methods may be sub-optimal and introduce noisy data such as black screens and blurry frames into the final video representation.

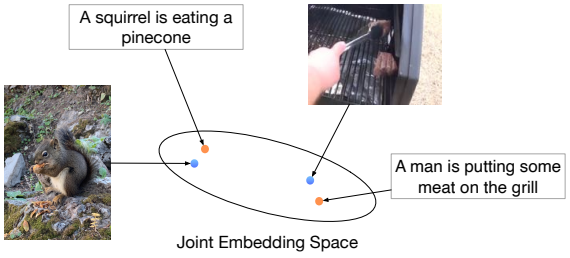


Figure 1. An example for multimodal joint representation. Blue dots and orange dots represent the vector representations for images and textual queries, respectively. In the joint embedding space, semantically similar content are closer in distance.

Motivated by this, we propose mmSampler, a learning-based approach to select *salient frames* from videos for the downstream video retrieval task. We design a policy network consisting of a lightweight feature extractor and a tempo-

[*]Equal contribution [1]Samsung AI Centre, Toronto, Canada. Correspondence to: Zhiming Hu <zhiming.hu@samsung.com>.

Proceedings of the 5th MLSys Conference, Santa Clara, CA, USA, 2022. Copyright 2022 by the author(s).

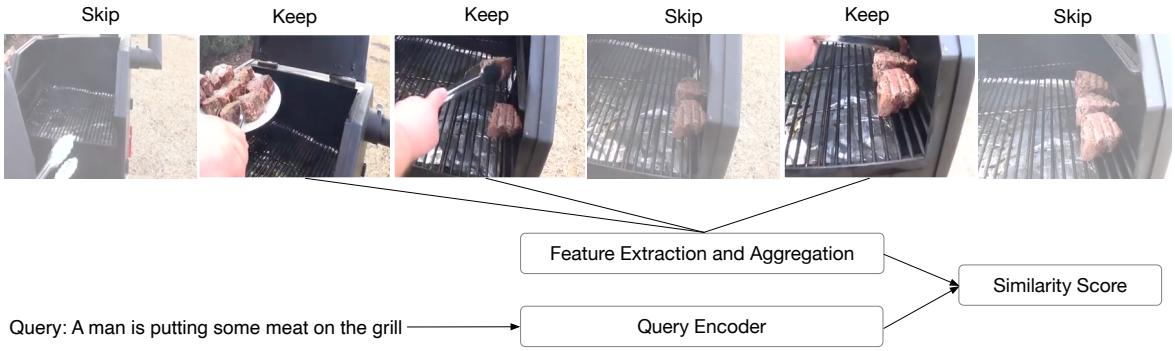


Figure 2. A motivating example for video-text matching in the inference stage. As an illustration, we start with six uniformly sampled frames. Instead of using all the frames in the downstream retrieval task, mmSampler will take a glance at the frames using a lightweight policy network and make the decision of which frames to keep independently of the textual query. Afterward, the visual features of the selected frames are extracted using a much more expensive image encoder, which are further aggregated to form the video embedding. Finally, both the video and query embeddings are projected to a common embedding space for similarity score calculation.

ral modelling module to decide which frames to preserve. Only the salient frames are processed by the computationally expensive video retrieval model, while redundant and uninformative frames are discarded. Fig. 2 illustrates a motivating example. By adaptively sampling and processing only a subset of salient frames, we can significantly reduce the computational overhead without sacrificing the video retrieval performance.

The key contributions of our paper are three-fold. First, to the best of our knowledge, mmSampler is the first work that aims to optimize the computational efficiency of multimodal video retrieval systems at runtime. Second, we propose a novel and lightweight approach to adaptively select salient frames for efficient video retrieval. We train our policy network end-to-end by adopting the Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) to resolve the non-differentiability problem arising from the discrete action space. Finally, extensive experimental results on several benchmarking datasets, including ActivityNet, DiDeMo and MSRVTT, have shown substantial benefits of mmSampler in saving as much as 43% GFLOPs per video while improving the retrieval performance.

2 Background and Motivation

This section provides background knowledge on multimodal video-text matching, followed by the motivation for a content-based frame sampler.

2.1 Multimodal Video-Text Matching

Generally, video-text matching learns joint multimodal representations for both videos and textual queries in a shared embedding space. In the joint space, semantically similar content are closer in distance, as depicted in Fig. 1.

Existing video-text retrieval frameworks either introduce cross-modal attention or embed the modalities separately to learn the final video-text representations. In the former approach, the video and text data are passed into a single encoder block (*e.g.*, a transformer architecture) to capture cross-modal alignment (Chen et al., 2018). Since the two modalities are tightly coupled, at retrieval time, an input query needs to be passed into the encoder along with a video to compute their similarity. This process repeats for *all* the candidate videos to return the best match(es), which is both inefficient and unscalable at inference time.

Alternatively, we can design *separate* encoders for each modality (Bain et al., 2021; Luo et al., 2021). The mapping of embeddings into a common multimodal space is learned using a contrastive loss (Radford et al., 2021) or a triplet loss (Liu et al., 2019). When adopting this design for video retrieval, the videos can be embedded offline, independent of the textual query. At retrieval time, an incoming search query is embedded and compared with the cached video embeddings, which is highly scalable. Therefore, we adopt this approach in our work.

2.2 Choosing Frames for Video Representation

Existing work on video-text retrieval generally processes all the frames (Miech et al., 2018; Liu et al., 2019) or a subset of frames in a video by uniform sampling (Bain et al., 2021; Luo et al., 2021). As both approaches are frame-agnostic, they may capture redundant frames or non-informative frames such as black screens and blurred frames. The presence of such frames may incur unnecessary computational overhead or hurt the video retrieval performance.

In the domain of action recognition, it has been shown that efficient frame sampling techniques (Korbar et al., 2019; Meng et al., 2020; Gao et al., 2020) can achieve both compu-

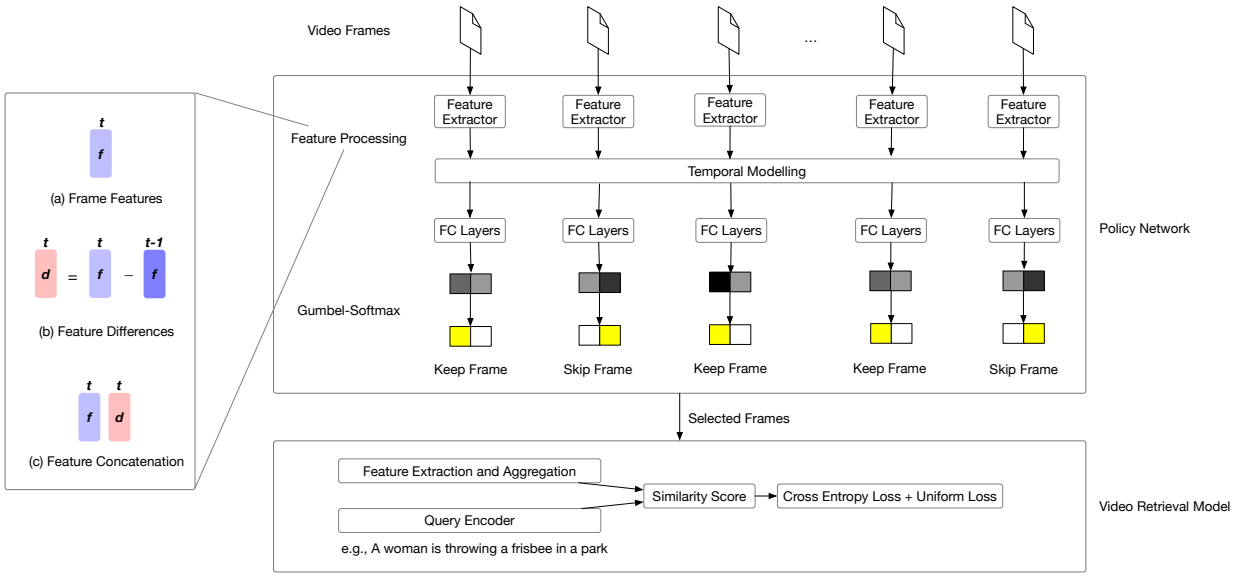


Figure 3. System overview for mmSampler. mmSampler uses a policy network to pick which frames to keep for retrieval. Frame features are first extracted using a lightweight feature extractor and passed into a temporal modelling module. We investigate three types of feature processing methods, namely frame features, feature differences, and feature concatenation. The output hidden states of the temporal modelling module are passed to a few fully connected (FC) layers to generate the logits for the frame actions. We use Gumbel-Softmax to generate the discrete action (keep or skip) for each frame given the logits. The selected frames are processed again with a more expensive image encoder. The visual features are aggregated into a video embedding and compared with the query embedding.

tational savings and improvements in classification accuracy by identifying and processing the salient frames/clips. A natural question to ask is could we achieve similar effects in multimodal video-text retrieval? In this work, we explore an adaptive sampling strategy to select salient frames from a diverse range of videos characterized by free-form natural language queries as opposed to videos represented by a fixed number of pre-determined action labels. In the following sections, we discuss our design in more detail and show improved retrieval performance with significant computational savings.

3 System Design

Videos contain a collection of unequally informative and important frames. We outline the technical design of mmSampler in this section and describe how it selectively chooses frames to reduce the video embedding overhead while improving the video-text retrieval performance.

3.1 System Overview

We propose mmSampler, an adaptive sampler to select salient video frames for efficient video-text retrieval. The network architecture is shown in Fig. 3. mmSampler learns a policy network to select important video frames to pass into a multimodal model for the downstream retrieval task.

Given an input video, we first uniformly sample a fixed number of frames. At each time step, we extract features for the current frame using a lightweight feature extractor and a temporal modelling module. The output representations are used as a signal to decide whether to keep or skip the observed frame for retrieval (**Sec. 3.2**). We also explore three different feature processing methods to capture appearance and/or motion information in videos (**Sec. 3.3**). Since the sampling decision is discrete and non-differentiable, we adopt the Gumbel-Softmax trick (Jang et al., 2016) and jointly train the policy network end-to-end with the downstream retrieval model in a differentiable manner (**Sec. 3.4**). The selected frames are passed into the multimodal retrieval model along with the textual query to determine a similarity matching score for video-text retrieval (**Sec. 3.5**). We train mmSampler using two specially designed losses to optimize retrieval performance and encourage frame skipping (**Sec. 3.6**).

3.2 Learning an Adaptive Policy Network

mmSampler learns to adaptively select salient frames using a policy network to pass into the downstream model for retrieval. The policy network is a neural network that defines how the actions are sampled at a given time. It is composed of three main components: a feature extractor, a temporal modelling module, and a frame selection module. We uniformly sample a fixed number frames from each video and decide whether to keep or skip each frame. The

policy network is guided by the textual queries at training time to identify salient frames specific to a dataset. However, at inference time, the frame sampling process is done independently of the input query, which is highly efficient.

We use a lightweight pretrained 2D CNN as a visual feature extractor by removing the final fully connected classification layer. We select a CNN that is considerably lighter in cost compared to the downstream multimodal model, adding negligible overhead to the final computational cost.

To capture temporal relations in a video, we pass the frame features into a temporal modelling module. This module can be a recurrent neural network such as an LSTM, or a lightweight 1-layer transformer network. An LSTM takes the current frame features and the previous states at each time step to generate new hidden and cell state vectors. The output hidden state of the LSTM encodes information from both the current frame and the history. A transformer can process the sequence of frames in parallel. The frame features are first projected to the same dimension as the transformer hidden size via a fully connected layer. Next, a position encoding is added to the features to retain the positional information of the input. The transformer outputs a new set of embeddings that captures the relation of a frame and every other frame in the sequence through its self-attention mechanism. The output hidden states of the temporal module are then passed into the frame selection module, which consists of a few fully connected layers to generate logits for our action space and the Gumbel-Softmax trick. These logits represent unnormalized probabilities for the discrete actions of keeping or skipping the frame.

3.3 Feature Processing

Inspired by action recognition works that adopt two-stream networks (Simonyan & Zisserman, 2014; Feichtenhofer et al., 2016) to capture both appearance and motion information using 2D CNNs, we propose three efficient methods to process the visual features to learn spatial and/or temporal dependencies in a video.

The first method uses the frame features directly, as shown in Fig. 3(a). At each time step t, the policy network first extracts visual features from the video frame using a lightweight feature extractor, then passes the features into the temporal modelling module with no additional change. Intuitively, this approach captures the spatial information in a video, which learns about the appearance of objects in a scene. By operating on CNN features, the sampler can learn to recognize uninformative scenes or objects in a frame that are specific to a downstream dataset.

The second method applies a difference operator, which computes the differences between the current frame feature and the previous frame feature. An overview is presented in Fig. 3(b). The intuition behind this method is that it uses visual feature differences to model motion relations between two frames, similar to optical flow. By learning the temporal dependencies between frames, the sampler can effectively remove some inter-frame redundancies without degrading the retrieval performance.

Lastly, we consider an integration of the first two methods, as illustrated in Fig. 3(c). Here, we apply a concatenation operator that takes in both the unprocessed frame features and the feature differences to form a 1D fixed-size vector. Inspired by the concept of two-stream networks in action recognition, this approach considers both the appearance and motion information in the final decision-making process. With the additional information, the sampler can learn to skip uninformative frames as well as some highly redundant frames to reduce the overall computational overhead and improve the retrieval performance.

3.4 Gumbel-Softmax Sampling

The objective of the policy network is to decide which frame to keep or skip. However, since the action space is discrete, the gradient for such an operation is undefined and unsuitable for backpropagation. A common workaround is to adopt policy-gradient methods in reinforcement learning such as the REINFORCE (Glynn, 1990; Williams, 1992) algorithm. However, such methods may have large variance in gradient estimates, leading to slow convergence (Zhao et al., 2011). In this work, we leverage the Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) to bypass the non-differentiability nature of the discrete sampling space and enable efficient end-to-end learning of our system.

At each time step t, we output a vector of logits $z \in \mathcal{R}^n$ where $n = 2$ is the size of the action space (keep action or skip action). The logits are passed to a softmax activation function to generate the probabilities p for each action in the sampling space. The probability for the i^{th} action follows a categorical distribution π_t,

$$\pi_t = \left\{ p_i \mid p_i = \frac{\exp(z_i)}{\sum_{j=0}^{n-1} \exp(z_j)} \right\}, i \in [0, \cdots, n-1] \tag{1}$$

We reparametrize the categorical distribution using the Gumbel-Max trick (Jang et al., 2016), which perturbs the probabilities by adding a standard Gumbel noise $G_i = -\log(-\log U_i)$ where U_i is drawn i.i.d. from a uniform distribution. In the forward pass, we sample an action $\hat{p}$ from the reparametrized distribution where the sampled actions are distributed according to π_t:

$$\hat{p} = \arg\max_i (\log p_i + G_i) \tag{2}$$

However, since the $\arg\max$ operator in Eq. (2) is still non-differentiable, we adopt the Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) to approximate the $\arg\max$ operator with the continuous softmax function as follows to allow for backpropagation:

$$q_i = \frac{\exp((\log p_i + G_i)/\tau)}{\sum_{j=0}^{n-1} \exp((\log p_j + G_j)/\tau)}, i \in [0, \cdots, n-1] \tag{3}$$

Here, q_i is the probability of choosing the i^{th} action and τ is a temperature parameter that controls how close the Gumbel-Softmax distribution is to the original categorical distribution. As τ approaches zero, the samples drawn from the Gumbel-Softmax distribution become one-hot. In practice, τ is initially set to a fixed value and gradually annealed down to a small non-zero temperature during training.

By leveraging the Gumbel-Softmax sampling technique, we can still sample from the true categorical distribution in the forward pass, while also allowing the network to be optimized via backpropagation.

3.5 Downstream Retrieval Model

We build a video-text retrieval model on top of the pre-trained image-text model CLIP (Radford et al., 2021). The retrieval model is finetuned with the policy network during training. Unlike prior work that processes all the uniformly sampled frames in a video for retrieval, we only use the frames selected by our policy network. The chosen frames are embedded by the finetuned CLIP visual encoder and aggregated via mean pooling to obtain the final video representation. The text representation is generated directly using the finetuned CLIP text encoder. Both the visual and text encoders already contain linear projection layers to map the embeddings to a common multimodal space. Given B video-text pairs during training, we compute B^2 cosine similarity scores between each video and text data point, where we seek to maximize the similarity scores of the B correct pairs and minimize the remaining $(B^2 - B)$ scores to learn an effective multimodal joint representation.

3.6 Loss Functions

While we do not have ground truth labels (*i.e.*, optimal action decisions) for the sampler to follow, we can alternatively supervise the policy network using a retrieval loss and a uniform loss. The retrieval loss is a symmetric cross entropy loss introduced in CLIP (Radford et al., 2021) to help maximize the cosine similarity scores between the positive image-text pairs and minimize the scores of the negative pairs. We adopt this loss to learn accurate video and text representations for retrieval, wherein the video representation is simply an aggregation of the individual frame features:

$$\mathcal{L}_{v2t} = -\frac{1}{B}\sum_{x=0}^{B-1} \log \frac{\exp(s_{x,x})}{\sum_{y=0}^{B-1}\exp(s_{x,y})}, \tag{4}$$

$$\mathcal{L}_{t2v} = -\frac{1}{B}\sum_{x=0}^{B-1} \log \frac{\exp(s_{x,x})}{\sum_{y=0}^{B-1}\exp(s_{y,x})}, \tag{5}$$

$$\mathcal{L}_r = \frac{1}{2}(\mathcal{L}_{v2t} + \mathcal{L}_{t2v}) \tag{6}$$

Here, B is the batch size, and $s_{x,y}$ is the cosine similarity between the x^{th} video and the y^{th} caption. $s_{x,x}$ represents the similarity score of a positive video-text pair.

By using only the retrieval loss $\mathcal{L}_r$, the model would be inclined to use almost all the frames to maximize the retrieval performance. Therefore, we also adopt the uniform loss in AR-Net (Meng et al., 2020) to encourage the policy network to sample different actions:

$$c_i = \frac{1}{B \times T}\sum_{b=0}^{B-1}\sum_{t=0}^{T-1} \mathbb{1}(a_{b,t} = i) - \frac{1}{n}, i \in [0, \cdots, n-1] \tag{7}$$

$$\mathcal{L}_u = \|\boldsymbol{c}\|_2 \tag{8}$$

Here, n is the total number of actions. $\|\cdot\|_2$ represents the L2 norm, and $\mathbb{1}(\cdot)$ is the indicator function. $\mathbb{1}(a_{b,t} = i)$ equals 1 if the selected action at time step t for the b^{th} sample video equals to the i^{th} action in the action space; otherwise it is 0. The term c_i compares the frequency of picking the i^{th} action in a batch with the expected frequency. The loss acts as a regularizer to penalize actions whose sampling frequency does not meet the expected frequency. If we only adopt this loss, it would promote a balanced policy across the actions.

The final training objective function for mmSampler is a combination of the two losses:

$$\mathcal{L} = w_r * \mathcal{L}_r + w_u * \mathcal{L}_u, \tag{9}$$

where w_r and w_u are the weights for the two losses, respectively. We set the weights empirically in our experiments.

4 Evaluation

In this section, we evaluate mmSampler in terms of retrieval performance and computational efficiency on several video datasets. We aim to answer the following questions:

- What is the video retrieval performance of mmSampler compared with SOTA approaches? Which feature processing technique(s) is the most effective?

- How much computational overhead does mmSampler save compared to uniform sampling?
- How do we optimize for the trade-off between retrieval performance and efficiency?
- Can mmSampler adapt to different visual backbones?
- What insights can be drawn from the sampling behaviour?

4.1 Experimental Setup

4.1.1 Datasets

We validate the performance of mmSampler on video-to-text and text-to-video retrieval on three benchmark datasets: ActivityNet, DiDeMo and MSRVTT.

ActivityNet Captions (Krishna et al., 2017) consists of 20,000 videos, where on average each is described by 3.65 temporally annotated sentences. Following prior work (Liu et al., 2019; Luo et al., 2021), we perform paragraph retrieval by concatenating the sentences in each video into one query for retrieval. We evaluate ActivityNet on the 'val1' split, which contains 10,009 train and 4,917 test videos.

DiDeMo (Hendricks et al., 2017) consists of 10,464 videos where each video is annotated with 3 to 5 temporally localized sentences. In total, there are 40,543 descriptions. The videos are split into 8,395 train, 1,065 validation, and 1,004 test. Following existing work (Luo et al., 2021; Fang et al., 2021), we perform paragraph retrieval.

MSRVTT (Xu et al., 2016) contains 10,000 trimmed YouTube video clips, each described by 20 captions. Evaluation is done on the '1k-A' split (9,000 train, 1,000 test) proposed by JSFusion (Yu et al., 2018). The test set contains 1,000 video-text pairs, in which we follow the standard retrieval evaluation procedure.

4.1.2 Evaluation Metrics

We evaluate video and text retrieval using standard retrieval metrics: recall at rank K (R@K) for K=$\{1,5,10\}$, median rank (MdR), and mean rank (MnR). R@K indicates the proportion of cases where the ground-truth caption/video is in the top-K retrieved items. For ActivityNet, K=50 is used in lieu of K=10 following prior work. For MdR and MnR, lower ranks correspond to better performance. We also report the number of giga-floating-point operations (GFLOPs) used to process each video as a measure for computational complexity.

4.1.3 Implementation Details

The policy consists of a feature extractor and a single-layer temporal modelling module (i.e., LSTM, transformer) with 512 hidden units. For the 1-layer transformer block, we set the number of self-attention heads to 8. The feature extractor is a frozen MobileNetV2 (Sandler et al., 2018) model pretrained on ImageNet (Deng et al., 2009). We explore three different feature processing techniques, namely frame features, feature differences, and feature concatenation, which we respectively denote as '**Ours**-frame', '**Ours**-diff', and '**Ours**-concat' in the results. If not otherwise specified, the default setting uses frame features as input and the 1-layer transformer block as the temporal modelling module.

We use two fully connected layers to map the temporal modelling module's hidden states to the logits for the sampling actions. The hidden layer dimension for the fully connected layers is 1024. The temperature parameter τ for the Gumbel-Softmax distribution is initially set to 5.0 and gradually annealed down using exponential decay with a decay constant of -0.045. By default, the first frame is always selected to ensure that the video representation is a valid vector.

The downstream multimodal retrieval model is initialized as a pretrained CLIP (ViT-B/32 model) (Radford et al., 2021), which is jointly trained with the policy network. If not otherwise stated, we train the model end-to-end for 30 epochs using the Adam optimizer with a learning rate of 1e-7 for the multimodal model parameters and 1e-4 for the remaining parameters. We train DiDeMo for only 5 epochs due to fast training convergence.

In terms of the input sequence lengths to CLIP's multimodal transformer encoders, we limit the caption token length to 32 and the number of sampled frames to 16 for MSRVTT. For ActivityNet and DiDeMo, the caption token length and the number of frames are 64 and 32, respectively. The frames are obtained by uniformly sampling the videos at their native frame rates. In our experiments, we set the weights for the losses w_r and w_u to be 1 and 0.03, respectively. All experiments are carried out on 4 NVIDIA Tesla A100 GPUs.

4.1.4 Baselines

We compare mmSampler with many prior state-of-the-art (SOTA) work on video retrieval including Collaborative Experts (Liu et al., 2019), Multi-Modal Transformer (Gabeur et al., 2020) and ClipBERT (Lei et al., 2021). Furthermore, we showcase retrieval and GFLOP numbers for finetuning the multimodal video retrieval model without the policy network. This baseline uses all N uniformly sampled frames, whereas mmSampler previews the same set of frames and selects only the salient frames for retrieval. In essence, the no-policy baseline has the same set-up as the CLIP4Clip-meanP (Luo et al., 2021) model, which finetunes CLIP's ViT-B/32 model on video datasets and applies mean pooling to the frame features to generate the final video representation. We denote the no-policy baseline as CLIP4Clip (Luo et al., 2021) (U, N), where U represents uniform sampling and N is the number of sampled frames per video.

Note that our baseline numbers for CLIP4Clip are from our reimplementation to ensure a fair comparison. Here, we outline some implementation differences. First, CLIP4Clip uses videos sampled at 1 fps, which may lead to instances where a sampled video contains less than N frames. In our work, we enforce consistency across videos by using the native video sampling rates such that each video has at least N frames. Second, we use 32 frames for ActivityNet and DiDeMo instead of 64 because we did not see substantial improvement from doubling the number of frames. Third, when training MSRVTT, we randomly sample one caption per video to use in training every epoch following existing work (Liu et al., 2019). This differs from CLIP4Clip, which is trained by using all the video-caption pairs every epoch. We observe that our training strategy yields better performance and does not overfit quickly to the training set.

Besides CLIP4Clip (Luo et al., 2021), we also compare with the concurrent work CLIP2Video (Fang et al., 2021). As their training code is unavailable, we evaluate their checkpoint on our set of extracted frames. Following the setting in their work, the model is validated on 12 uniformly sampled frames per video for MSRVTT.

4.2 Experimental Results

4.2.1 Comparison to Video-Text Retrieval Baselines

Tables 1-3 list the text-to-video and video-to-text retrieval results of mmSampler and prior work on ActivityNet, DiDeMo and MSRVTT datasets. We also compare the GFLOP usage per video for our model and the no-policy CLIP4Clip baseline. Across the three datasets, mmSampler achieves SOTA retrieval performance over the baselines in most cases while showing substantial reduction in computational requirements.

At first glance, it can be observed that CLIP-based approaches such as CLIP4Clip and CLIP2Video outperform other baselines by a significant margin. One key reason for such performance boost is that CLIP is pretrained on a large-scale image-text dataset (400 million pairs) to learn generalized image and text embeddings. Most prior arts are either trained on the downstream dataset directly or first pretrained on smaller image or video datasets such as HowTo100M (Miech et al., 2019).

For ActivityNet in Table 1, mmSampler exceeds the no-policy results in most of the retrieval metrics, showing as much as 1.4% and 1.9% improvement in text-to-video and video-to-text retrieval R@1, respectively. mmSampler also reduces the number of frames used by almost 50% compared to CLIP4Clip. Since fewer frames are selected and passed into the heavyweight CLIP visual encoder, mmSampler reduces the computational overhead by 43% for the three feature processing methods.

Similarly, DiDeMo results in Table 2 also show consistent improvement over the CLIP4Clip baseline, where mmSampler exceeds the baseline for all R@K (K=$\{1,5,10\}$) metrics. On average, we use 33% to 41% less GFLOPs and 13 to 15 fewer frames per video.

Lastly, for MSRVTT, mmSampler on average selects 10 to 11 frames per video out of 16 uniformly sampled frames, offering 24-32% savings in GFLOPs compared to the baseline. For the retrieval performance, all three feature processing methods outperform the no-policy baseline, yielding as high as a 1.5% and 2.3% improvement in text-to-video and video-to-text retrieval R@1 under the frame feature setting.

We note that although CLIP2Video (Fang et al., 2021) shows competitive retrieval performance on the MSRVTT dataset, the model incurs greater computational overhead compared to CLIP4Clip and mmSampler due to the addition of a cross-modal alignment module and several temporal transformer blocks. The key objective of mmSampler is to reduce the number of frames processed by the heavyweight retrieval model while improving the retrieval performance. It is designed to be general and can be plugged into different retrieval backbones (See more results in Table 4). We show the effectiveness of mmSampler with CLIP4Clip in this paper. Evaluating mmSampler with CLIP2Video (Fang et al., 2021) could be an interesting future direction.

In terms of the three feature processing methods, **Ours**-diff is the most computationally efficient method overall. This suggests that by incorporating motion-based features, we can effectively remove more redundant frames from the video compared to appearance features. We also observe that **Ours**-diff generally performs worse than **Ours**-frame, especially for MSRVTT, suggesting that spatial context might be more informative than motion dynamics for some datasets. **Ours**-concat shows the best overall performance for ActivityNet, emphasizing the benefits of both static appearance and dynamic motion information in a video retrieval setting. By leveraging both streams, mmSampler can effectively filter out uninformative and some redundant frames to arrive at a more robust and accurate video representation.

In summary, mmSampler achieves strong retrieval performance on all three datasets by sampling and further processing only the salient frames for retrieval. Not only have we improved the recall at rank K (R@K), mmSampler also yields lower median rank (MdR) and mean rank (MnR) compared to the no-policy baseline, which capture the overall retrieval performance for all the videos in a dataset. By using only a subset of the original frames, we greatly reduce the computational overhead associated with processing the frames. We demonstrate that more frames may not always be better given that certain frames are uninformative and may hinder the learning of an effective video representation.

Table 1. Text-to-video retrieval, video-to-text retrieval, and average GFLOPs per video on ActivityNet. (U, N) and (P, N) represent uniform sampling and policy-based sampling N frames per video, respectively. For example, (P, 16.25) represents sampling 16.25 frames per video on average with the learned policy.

		Text $\Longrightarrow$ Video					Video $\Longrightarrow$ Text				
Methods	GFLOPs/v	R@1	R@5	R@50	MdR	MnR	R@1	R@5	R@50	MdR	MnR
CE (Liu et al., 2019)	-	18.2	47.7	91.4	6	23.1	17.7	46.6	90.9	6	24.4
TT-CE+ (Croitoru et al., 2021)	-	23.5	57.2	96.1	4	13.7	23.0	56.1	95.8	4	-
MMT-PT (Gabeur et al., 2020)	-	28.7	61.4	94.5	3.3	16.0	28.9	61.1	94.3	4	17.1
SSB-PT (Patrick et al., 2020)	-	29.2	61.6	94.7	3	-	28.7	60.8	94.8	**2**	-
ClipBERT (Lei et al., 2021)	-	21.3	49.0	-	6	-	-	-	-	-	-
MDMMT (Dzabraev et al., 2021)	-	20.1	45.1	-	7	7.8	-	-	-	-	-
CLIP4Clip (Luo et al., 2021) (U, 32)	141.2	41.3	72.3	**98.0**	**2**	7.5	42.1	74.0	**98.3**	**2**	7.0
Ours-frame (P, 15.9)	80.4	42.0	72.4	97.8	**2**	7.4	43.7	74.4	98.0	**2**	7.0
Ours-diff (P, 15.8)	**79.9**	42.2	72.2	97.7	**2**	7.6	43.0	74.0	97.9	**2**	7.2
Ours-concat (P, 16.0)	80.9	**42.7**	**72.9**	97.9	**2**	**7.2**	**44.0**	**74.5**	98.1	**2**	**6.8**

Table 2. Text-to-video retrieval, video-to-text retrieval, and average GFLOPs per video on DiDeMo. (U, N) and (P, N) represent uniform sampling and policy-based sampling N frames per video, respectively. For example, (U, 32) represents uniformly sampling 32 frames.

		Text $\Longrightarrow$ Video					Video $\Longrightarrow$ Text				
Methods	GFLOPs/v	R@1	R@5	R@10	MdR	MnR	R@1	R@5	R@10	MdR	MnR
CE (Liu et al., 2019)	-	16.1	41.1	-	8.3	43.7	15.6	40.9	-	8.2	42.4
TT-CE+ (Croitoru et al., 2021)	-	21.6	48.6	62.9	6	-	21.1	47.3	61.1	6.3	-
Frozen (Bain et al., 2021)	-	34.6	65.0	74.7	3	-	-	-	-	-	-
ClipBERT (Lei et al., 2021)	-	20.4	48.0	60.8	6	-	-	-	-	-	-
CLIP4Clip (Luo et al., 2021) (U, 32)	141.2	40.7	68.9	79.1	**2**	18.6	41.0	68.9	79.2	**2**	12.2
Ours-frame (P, 19.2)	94.9	41.4	**70.1**	**80.0**	**2**	**18.3**	41.8	70.9	**80.5**	**2**	**11.0**
Ours-diff (P, 16.8)	84.4	41.3	69.0	80.3	**2**	18.7	**42.0**	69.8	79.6	**2**	11.3
Ours-concat (P, 16.7)	**84.0**	**41.6**	70.0	79.7	**2**	**18.3**	41.7	**71.2**	79.3	**2**	11.4

4.2.2 *Retrieval Performance and Efficiency Trade-off*

As discussed in Sec. 3.6, our overall loss function given by Eq. 9 consists of a retrieval loss and a uniform loss, which directly affect the retrieval performance and computational efficiency. We can toggle the weights for the individual losses to observe different retrieval performance and efficiency trade-offs.

In Table 4, we provide several combinations of the loss weights w_r and w_u as well as the corresponding retrieval performance and GFLOPs usage per video. If only the retrieval loss is used ($w_r = 1, w_u = 0$), the policy should be inclined to keep almost all the frames for retrieval in an attempt to maximize performance, which is similar to the CLIP4Clip baseline. Interestingly, we observe that the sampler uses 4 fewer frames per video on average while achieving better performance compared to the baseline. This suggests that using all the frames does not necessarily correlate with the best performance.

Adding uniform loss into the overall objective function can encourage the policy to start learning to skip frames. Based on our observations, setting a high weight for the retrieval loss ($w_r = 1$) and adding a small weight for the uniform loss ($w_u = 0.03$) yields the best retrieval performance and efficiency trade-off across all the datasets. Setting the weight for the uniform loss w_u to be too large ($w_u = 0.3$) may drive the policy to skip too aggressively and in turn harm the retrieval performance.

4.2.3 *Ablation Studies*

Table 4 presents two other ablation studies we have conducted, specifically the benefits of the temporal modelling module and the performance of adopting another pretrained, more heavyweight CLIP visual encoder[1].

Temporal Modelling Module We investigate the benefits of different temporal modelling modules to capture temporal dependencies. We observe in Table 4 that the 1-layer transformer architecture outperforms both the LSTM and bi-LSTM modules in both performance and computation efficiency. Furthermore, the performance of using a temporal modelling module is better than feeding the input features directly to the fully connected layers.

[1]More results are presented in the Appendix.

Table 3. Text-to-video retrieval, video-to-text retrieval, and average GFLOPs per video on the 1k-A split of MSRVTT. (U, N) and (P, N) represent uniform sampling and policy-based sampling N frames per video, respectively. For example, (U, 16) represents uniformly sampling 16 frames. CLIP2Video (Fang et al., 2021) results are obtained from evaluating their checkpoint on 12 uniformly sampled frames per video with frames extracted at the native frame rate.

		Text $\Longrightarrow$ Video					Video $\Longrightarrow$ Text				
Methods	GFLOPs/v	R@1	R@5	R@10	MdR	MnR	R@1	R@5	R@10	MdR	MnR
JSFusion (Yu et al., 2018)	-	10.2	31.2	43.2	13	-	-	-	-	-	-
CE (Liu et al., 2019)	-	20.9	48.8	62.4	6	28.2	20.6	50.3	64.0	5.3	25.1
TT-CE+ (Croitoru et al., 2021)	-	29.6	61.6	74.2	3	-	32.1	62.7	75.0	3	-
ActBERT (Zhu & Yang, 2020)	-	8.6	23.4	33.1	36	-	-	-	-	-	-
ClipBERT (Lei et al., 2021)	-	22.0	46.8	59.9	6	-	-	-	-	-	-
HowTo100M (Miech et al., 2019)	-	14.9	40.2	52.8	9	-	16.8	41.7	55.1	8	-
MMT-PT (Gabeur et al., 2020)	-	26.6	57.1	69.6	4	24.0	27.0	57.5	69.7	3.7	21.3
SSB-PT (Patrick et al., 2020)	-	30.1	58.5	69.3	3	-	28.5	58.6	71.6	3	-
Frozen (Bain et al., 2021)	-	31.0	59.5	70.5	3	-	-	-	-	-	-
Straight-CLIP (Portillo-Quintero et al., 2021)	-	31.2	53.7	64.2	4	-	27.2	51.7	62.6	5	-
MDMMT (Dzabraev et al., 2021)	-	38.9	69.0	79.7	**2**	16.5	-	-	-	-	-
CLIP2Video (Fang et al., 2021)	-	**44.1**	**71.7**	**81.6**	**2**	**14.1**	43.4	71.3	82.0	**2**	**10.0**
CLIP4Clip (Luo et al., 2021) (U, 16)	70.6	42.2	68.7	79.2	**2**	16.5	42.1	70.4	81.2	**2**	11.7
Ours-frame (P, 11.0)	53.8	43.7	71.2	79.8	**2**	15.6	**44.4**	71.5	82.1	**2**	10.6
Ours-diff (P, 9.7)	**47.8**	42.8	70.6	80.6	**2**	15.6	43.9	71.3	**82.4**	**2**	10.9
Ours-concat (P, 10.1)	49.7	43.6	70.6	80.1	**2**	15.8	44.3	**72.1**	82.1	**2**	11.2

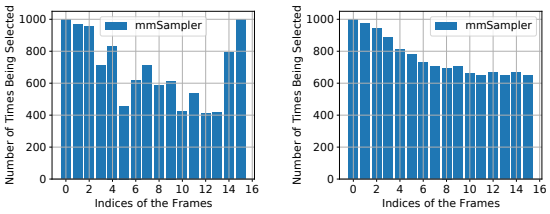
Table 4. Ablation studies on MSRVTT 1k-A split for different temporal modelling modules, loss weights, and backbones. w_r and w_u are the weights for the retrieval loss and uniform loss, respectively. The row in bold highlights mmSampler results obtained under the default setting of using MobileNetV2 frame features.

					Text $\Longrightarrow$ Video			Video $\Longrightarrow$ Text		
Methods	Temporal Model	(w_r, w_u)	Backbone	GFLOPS/v	R@1	R@5	R@10	R@1	R@5	R@10
CLIP4Clip (Luo et al., 2021) (U, 16)	-	-	ViT-B/32	70.6	42.2	68.7	79.2	42.1	70.4	81.2
Ours-frame (P, 12.0)	Transformer	(1, 0.0)	ViT-B/32	58.2	43.0	70.4	81.2	43.0	69.7	82.0
Ours-frame (P, 11.0)	**Transformer**	**(1, 0.03)**	**ViT-B/32**	**53.8**	**43.7**	**71.2**	**79.8**	**44.4**	**71.5**	**82.1**
Ours-frame (P, 8.8)	Transformer	(1, 0.3)	ViT-B/32	44.1	43.2	71.1	78.8	43.1	71.4	81.6
Ours-frame (P, 12.2)	LSTM	(1, 0.03)	ViT-B/32	58.8	42.9	70.1	80.8	43.4	71.4	82.2
Ours-frame (P, 11.7)	Bi-LSTM	(1, 0.03)	ViT-B/32	56.7	42.0	71.2	80.0	43.6	71.4	81.8
Ours-frame (P, 11.9)	None	(1, 0.03)	ViT-B/32	57.6	42.6	69.4	80.3	42.0	71.3	81.3
CLIP4Clip (Luo et al., 2021) (U, 16)	-	-	ViT-B/16	281.2	44.5	71.2	80.6	44.0	72.1	81.9
Ours-frame (P, 11.5)	Transformer	(1, 0.03)	ViT-B/16	183.4	45.8	73.1	82.1	45.6	73.6	83.1

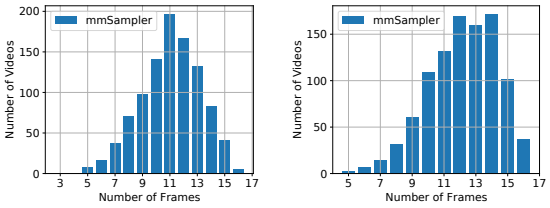
Different CLIP Backbone We also investigate the performance of another CLIP visual backbone, ViT-B/16. The last two rows of Table 4 present the ViT-B/16 baseline results for using uniformly sampled frames and the performance of our sampler trained on the same downstream model. We observe similar trends as the ViT-B/32 backbone. By adopting our content-aware sampler, we achieve better retrieval performance in all metrics while reducing the GFLOPs usage by 35%. We also see that the sampler achieves even better retrieval results using the ViT-B/16 backbone compared to mmSampler with the ViT-B/32 backbone. However, the new backbone requires a substantially higher amount of computation. In sum, the key takeaway is that mmSampler can be trained end-to-end with different video retrieval backbones to improve retrieval performance and save computation.

4.2.4 *Analysis of the Sampling Behaviours*

We present the distribution of the location of the sampled frames and the number of sampled frames per video on MSRVTT for both the 1-layer transformer and the LSTM temporal modelling modules in Fig. 4. Note that the first frame is always chosen by default so that at least one frame is picked per video. Fig. 4(a) demonstrates that the transformer network selects the first few frames and the last frame more frequently than the rest of the video frames. On the other hand, the LSTM module tends to select frames from the beginning. We conjecture that because the transformer has access to the global context in a video, it samples frames that are more distant or distributed to represent a video. As an LSTM processes the frames sequentially and only has access to historical context, it will select more frames from the beginning. The frames near the end could be redundant



(a) Location of selected frames using a 1-layer transformer (left) and an LSTM (right).



(b) Number of frames selected per video using a 1-layer transformer (left) and an LSTM (right).

Figure 4. Sampling behaviours of transformer and LSTM in the policy network. In Fig. 4(a), we observe the transformer picks more frames from the beginning and end of the videos, while an LSTM selects more frames from the beginning. In Fig. 4(b), we show the average number of frames being selected. mmSampler does not select too many or too few frames.

compared to what has already been viewed.

We also show the average number of frames selected by the mmSampler in Fig. 4(b). As shown by the distribution, the transformer selects fewer frames on average compared to the LSTM. In both cases, mmSampler would rarely select fewer than six frames or use all the frames. Skipping too many frames may harm the retrieval performance while keeping too many frames will increase the computational overhead. In our case, we encourage the policy to choose the skip action with the uniform loss.

4.2.5 *Sampler Generalizability*

We evaluate the extent of the frame sampler to generalize to unseen videos. Redundant and uninformative frames are inherent characteristics of many videos, but the uninformative content may be different across datasets. The results in the paper are produced by evaluating mmSampler on the same dataset on which it is trained. We test the generalizability of mmSampler by freezing the CLIP backbone and training only the weights of the policy network on ActivityNet, the largest of all three datasets. The same sampler weights are used to evaluate on other datasets. The results are presented in Table 5. We observe the benefit of the sampler on all datasets in both computational savings and performance enhancement, demonstrating the ability of the sampler to generalize to different datasets.

5 Limitations and Future Work

In this section, we discuss the limitations and potential future directions of mmSampler.

Multiple Modalities In this paper, we achieve strong video retrieval performance through the use of frame-level visual features alone. However, videos are more than a sequence of frames. They are rich in modalities such as audio, speech and motion, which have shown to be beneficial for improving the video retrieval performance (Liu et al., 2019; Miech et al., 2018). How to adapt mmSampler for other modalities to achieve even better video retrieval performance is a promising future research direction.

Specialized Video Sampler Some datasets such as MSRVTT (Xu et al., 2016) contain video clips spanning various domains, including but not limited to cartoons, movies, video games and cooking videos. The video frames are substantially different across domains. For instance, in some cooking videos, the scene is almost static across the entire video. On the other hand, multiple scene changes may be observed in some cartoon videos. Therefore, designing a specialized video sampler for individual video types may further improve the efficiency of video analytics without affecting the retrieval performance. Moreover, one may consider only sampling among keyframes (i.e., I-frames) in compressed videos instead of sampling across all frames, as inter-frames (i.e., P-frames or B-frames) may be computationally expensive to compute on mobile or edge devices.

Other Downstream Tasks While this work has only studied the benefit of a policy network in a video retrieval context, the sampler concept can also be tailored to other multimodal video understanding tasks, such as video summarization and video captioning. For video summarization, the task is to select representative frames for each video, which is also aligned with the goals in this work. For video captioning, it has been shown that using only the informative frames is beneficial (Chen et al., 2018); therefore, we may be able to adopt the sampler for this task. Overall, the key advantage of our design is that mmSampler is differentiable and can be trained end-to-end with the downstream task to achieve better performance.

6 Related Work

In this section, we discuss recent related work in three different research directions.

Natural Language-based Video Retrieval

Video retrieval has been actively researched in recent years. One approach involves extracting features from different video modalities such as RGB, motion, and audio, and design a multimodal fusion mechanism to map the modality representations to a shared embedding space (Miech et al.,

Table 5. Performance of a sampler trained on ActivityNet and evaluated on various datasets. The CLIP backbone is frozen in all cases.

				Text ⟹ Video			Video ⟹ Text		
Methods	Dataset	Temporal Model	GFLOPS/v	R@1	R@5	R@10	R@1	R@5	R@10
Frozen CLIP (Radford et al., 2021) (U, 32)	ActivityNet	-	141.2	19.9	44.7	57.8	18.8	42.9	56.5
Ours-frame (P, 24.7)	ActivityNet	Transformer	119.4	21.1	45.6	58.8	19.2	43.6	57.6
Ours-frame (P, 18.9)	ActivityNet	LSTM	93.5	21.2	46.0	58.7	19.5	44.0	57.9
Frozen CLIP (Radford et al., 2021) (U, 32)	DiDeMo	-	141.2	26.9	50.7	62.5	23.4	49.8	61.5
Ours-frame (P, 23.7)	DiDeMo	Transformer	115.1	27.6	51.3	62.0	25.1	50.2	61.9
Ours-frame (P, 19.3)	DiDeMo	LSTM	95.4	26.7	51.4	62.2	25.3	50.4	62.0
Frozen CLIP (Radford et al., 2021) (U, 32)	MSRVTT	-	141.2	30.2	54.1	63.0	26.1	51.5	62.6
Ours-frame (P, 24.3)	MSRVTT	Transformer	117.3	31.2	55.7	64.5	26.6	53.0	63.7
Ours-frame (P, 17.0)	MSRVTT	LSTM	85.2	31.3	54.9	63.5	27.1	52.0	63.6

2018; Liu et al., 2019; Gabeur et al., 2020). Recently, we have seen remarkable retrieval performance in both zero-shot and finetuned settings (Patrick et al., 2020; Bain et al., 2021; Dzabraev et al., 2021) by pretraining on large datasets such as HowTo100M (Miech et al., 2019). Such works reinforce the importance of pretraining on large amounts of data to learn more generalized video-text representations.

However, obtaining large-scale annotated video datasets is expensive. Therefore, an emerging line of work investigates using frame-level visual features alone (Portillo-Quintero et al., 2021; Luo et al., 2021; Fang et al., 2021) for the video representation instead of using an aggregation of features from different modalities. These works leverage the image encoder from CLIP (Radford et al., 2021), an image-text model pretrained on a large image-text dataset, to extract and aggregate frame-based features for the final video representation, achieving state-of-the-art retrieval performance.

Generally, video retrieval frameworks are optimized for accuracy and they extract features for all the frames or adopt uniform sampling, which could be inefficient or inject uninformative frames into the final embedding. Our policy is designed to sample only a subset of informative frames to use in the video-text retrieval model, dramatically saving the computational cost associated with utilizing all the frames.

Efficient Action Recognition

Action recognition has been extensively studied by the computer vision community. Most works mainly focus on designing powerful and deep networks to achieve state-of-the-art performance without taking into account the overall computational cost (Carreira & Zisserman, 2017; Feichtenhofer et al., 2018). Research has also been conducted on efficient action recognition, which aims to design more lightweight architectures (Tran et al., 2018; Lin et al., 2019; Tran et al., 2019) or devise intelligent sampling techniques to achieve better prediction accuracy (Korbar et al., 2019; Meng et al., 2020; Gao et al., 2020). For instance, SCSampler (Korbar et al., 2019) and Listen to Look (Gao et al., 2020) use audio as an additional modality to select salient clips for action recognition. AR-Net (Meng et al., 2020) proposes a policy network to decide which input resolution to use in the action recognition model on a per-frame basis. On the other hand, our proposed solution is targeted at a multimodal learning task where the natural language queries are free form and not bounded by a limited number of classes.

Efficient Video Analytics

Efficient video analytics (Kang et al., 2017; Hsieh et al., 2018; Kraft et al., 2019; Fu et al., 2019; Shen et al., 2019) is an important research problem in the computer vision and system communities. More specifically, MIRIS (Bastani et al., 2020) and ExSample (Moll et al., 2020) introduce efficient sampling algorithms for domain specific language (DSL)-based video analytics. Although these works also aim to process fewer frames while maintaining the same level of accuracy as dense sampling, our focus is on natural language-based video retrieval, where the query can be any free-form text.

7 Concluding Remarks

We propose a learning-based policy sampler, mmSampler, to select salient frames from a video for video-text retrieval. Existing works generally use the complete set of frames or uniformly sample a subset, which may not be efficient and may introduce uninformative frames into the final video representation. We design and train a lightweight policy network end-to-end with the multimodal retrieval model by adopting the Gumbel-Softmax trick. Experimental results on benchmark datasets including ActivityNet, DiDeMo and MSRVTT indicate the effectiveness of mmSampler. Our design shows improved retrieval performance and significant GFLOPs savings by as much as 43% per video.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. We are also grateful to Kosta Derpanis for the insightful discussions on the draft of the paper.

REFERENCES

Bain, M., Nagrani, A., Varol, G., and Zisserman, A. Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval. *arXiv preprint arXiv:2104.00650*, 2021.

Bastani, F., He, S., Balasingam, A., Gopalakrishnan, K., Alizadeh, M., Balakrishnan, H., Cafarella, M., Kraska, T., and Madden, S. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 1907–1921, 2020.

Carreira, J. and Zisserman, A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Chen, Y., Wang, S., Zhang, W., and Huang, Q. Less is More: Picking Informative Frames for Video Captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 358–373, 2018.

Croitoru, I., Bogolin, S.-V., Liu, Y., Albanie, S., Leordeanu, M., Jin, H., and Zisserman, A. TEACHTEXT: CrossModal Generalized Distillation for Text-Video Retrieval. *arXiv preprint arXiv:2104.08271*, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A Large-Scale Hierarchical Image Database. In *Proc. of IEEE CVPR*, pp. 248–255, 2009.

Dzabraev, M., Kalashnikov, M., Komkov, S., and Petiushko, A. MDMMT: Multidomain Multimodal Transformer for Video Retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3354–3363, 2021.

Fabian Caba Heilbron, Victor Escorcia, B. G. and Niebles, J. C. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

Fang, H., Xiong, P., Xu, L., and Chen, Y. CLIP2Video: Mastering Video-Text Retrieval via Image CLIP. *arXiv preprint arXiv:2106.11097*, 2021.

Feichtenhofer, C., Pinz, A., and Wildes, R. P. Spatiotemporal Residual Networks for Video Action Recognition. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 34763484. Curran Associates Inc., 2016. ISBN 9781510838819.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. SlowFast Networks for Video Recognition. *arXiv preprint arXiv:1812.03982*, 2018.

Fu, D. Y., Crichton, W., Hong, J., Yao, X., Zhang, H., Truong, A., Narayan, A., Agrawala, M., Ré, C., and Fatahalian, K. Rekall: Specifying Video Events using Compositions of Spatiotemporal Labels. *arXiv preprint arXiv:1910.02993*, 2019.

Gabeur, V., Sun, C., Alahari, K., and Schmid, C. Multimodal Transformer for Video Retrieval. In *European Conference on Computer Vision (ECCV)*, 2020.

Gao, R., Oh, T.-H., Grauman, K., and Torresani, L. Listen to Look: Action Recognition by Previewing Audio. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10457–10467, 2020.

Glynn, P. W. Likelihood Ratio Gradient Estimation for Stochastic Systems. *Communications of the ACM*, 33 (10):75–84, 1990.

Hendricks, L. A., Wang, O., Shechtman, E., Sivic, J., Darrell, T., and Russell, B. Localizing Moments in Video with Natural Language. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Hsieh, K., Ananthanarayanan, G., Bodik, P., Venkataraman, S., Bahl, P., Philipose, M., Gibbons, P. B., and Mutlu, O. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 269–286, 2018.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. Noscope: Optimizing Neural Network Queries over Video at Scale. *arXiv preprint arXiv:1703.02529*, 2017.

Korbar, B., Tran, D., and Torresani, L. SCSampler: Sampling Salient Clips from Video for Efficient Action Recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6232–6242, 2019.

Kraft, P., Kang, D., Narayanan, D., Palkar, S., Bailis, P., and Zaharia, M. Willump: A Statistically-Aware End-to-End Optimizer for Machine Learning Inference. *arXiv preprint arXiv:1906.01974*, 2019.

Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Carlos Niebles, J. Dense-Captioning Events in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 706–715, 2017.

Lei, J., Li, L., Zhou, L., Gan, Z., Berg, T. L., Bansal, M., and Liu, J. 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*, pp. 7331–7341, 2021.

Lin, J., Gan, C., and Han, S. TSM: Temporal Shift Module for Efficient Video Understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Liu, Y., Albanie, S., Nagrani, A., and Zisserman, A. Use What You Have: Video Retrieval using Representations from Collaborative Experts. In *arXiv preprint arxiv:1907.13487*, 2019.

Luo, H., Ji, L., Zhong, M., Chen, Y., Lei, W., Duan, N., and Li, T. CLIP4Clip: An Empirical Study of Clip for End to End Video Clip Retrieval. *arXiv preprint arXiv:2104.08860*, 2021.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv preprint arXiv:1611.00712*, 2016.

Meng, Y., Lin, C.-C., Panda, R., Sattigeri, P., Karlinsky, L., Oliva, A., Saenko, K., and Feris, R. AR-Net: Adaptive Frame Resolution for Efficient Action Recognition. In *European Conference on Computer Vision*, pp. 86–104. Springer, 2020.

Miech, A., Laptev, I., and Sivic, J. Learning a Text-Video Embedding from Incomplete and Heterogeneous Data. *arXiv:1804.02516*, 2018.

Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., and Sivic, J. Howto100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2630–2640, 2019.

Moll, O., Bastani, F., Madden, S., Stonebraker, M., Gadepally, V., and Kraska, T. ExSample: Efficient Searches on Video Repositories through Adaptive Sampling. *arXiv preprint arXiv:2005.09141*, 2020.

Patrick, M., Huang, P., Asano, Y. M., Metze, F., Hauptmann, A. G., Henriques, J. F., and Vedaldi, A. Support-Set Bottlenecks for Video-Text Representation Learning. *arXiv preprint arXiv:2010.02824*, 2020.

Portillo-Quintero, J. A. et al. A Straightforward Framework for Video Retrieval Using CLIP. In *Mexican Conference on Pattern Recognition*, pp. 3–12. Springer, 2021.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning Transferable Visual Models from Natural Language Supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Shen, H., Chen, L., Jin, Y., Zhao, L., Kong, B., Philipose, M., Krishnamurthy, A., and Sundaram, R. Nexus: A GPU Cluster Engine for Accelerating DNN-Based Video Analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pp. 322337, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368735. doi: 10.1145/3341301.3359658. URL https://doi.org/10.1145/3341301.3359658.

Simonyan, K. and Zisserman, A. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pp. 568–576. MIT Press, 2014.

Sovrasov, V. Flops Counter for Convolutional Networks in Pytorch Framework. https://bit.ly/3mtHX5J, 2021.

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 09–15 Jun 2019.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

Tran, D., Wang, H., Torresani, L., and Feiszli, M. Video Classification With Channel-Separated Convolutional Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning*, 8(3):229–256, 1992.

Xu, J., Mei, T., Yao, T., and Rui, Y. MSRVTT: A Large Video Description Dataset for Bridging Video and Language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5288–5296, 2016.

Yu, Y., Kim, J., and Kim, G. A Joint Sequence Fusion Model for Video Question Answering and Retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. Analysis and Improvement of Policy Gradient Estimation. In *NIPS*, pp. 262–270. Citeseer, 2011.

Zhu, L. and Yang, Y. Actbert: Learning Global-Local Video-Text Representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8746–8755, 2020.

A Artifact Appendix

A.1 Abstract

The artifact contains the instructions to prepare the datasets, setup the environment and run the scripts for both training and testing. Users should be able to reproduce all the experiments in the paper and appendix.

A.2 Artifact check-list (meta-information)

- **Data set:** MSRVTT (76GB), DiDeMo (676GB), ActivityNet (4.6TB)
- **Run-time environment:** Ubuntu 16.04 (or newer)
- **Hardware:** A machine with at least one Nividia GPU. The artifact was tested on 4 NVIDIA A100s.
- **Metrics:** Recall at rank K, GFLOPs per video
- **Output:** Log is both printed to console and saved
- **Experiments:** Text-to-video and video-to-text retrieval on several benchmarking datasets including ActivityNet, DiDeMo and MSRVTT.
- **How much disk space required (approximately)?:** Around 20GB for the artifact and trained models, excluding datasets
- **How much time is needed to complete experiments (approximately)?:** One day
- **Publicly available?:** No

A.3 Description

A.3.1 How delivered

We hope to release the code for the paper soon. The amount of disk space required after unpacking the artifact is under 100MB.

A.3.2 Hardware dependencies

A machine with at least one GPU of memory greater than 16 GB is recommended. However, if there is less GPU memory available, a smaller batch size can be used. As a reference, training MSRVTT with a batch size of 16 requires 14GB of GPU memory.

A.3.3 Software dependencies

A Linux machine with at least Ubuntu 16.04 is recommended. We require ffmpeg to be installed (`sudo apt-get install ffmpeg`) for frame extraction. It would be best for the experiments to be done in a virtual environment with Python 3, conda, and pip installed.

A.3.4 Data sets

We evaluate mmSampler on three datasets: MSRVTT, DiDeMo, and ActivityNet. MSRVTT is shared by the authors of Frozen in Time (Bain et al., 2021). The videos are approximately 6.3GB in size. DiDeMo can be downloaded from the original author's GitHub, either through AWS S3 or Google Drive. The videos take up 51GB of storage space. ActivityNet can be downloaded from the official website (Fabian Caba Heilbron & Niebles, 2015) by filling in a request form. The videos require 375GB of space. Refer to the `README.md` file for the specific links to the videos. The videos should all be put in the same directory. Do not separate them into sub-directories.

A.4 Installation

To conduct the experiments, users need to set up a new conda environment (or any virtual environment) on their machines and install the appropriate packages:

```
# source scripts/install_packages.sh
```

You may need to update `cudatoolkit=11.0` in the script with the appropriate CUDA version on your machine.

A.5 Experiment workflow

After all the videos for a dataset have been downloaded, extract the frames using the script `frame_extraction.py`.

The raw frames for MSRVTT, ActivityNet, and DiDeMo take up approximately 70GB, 625GB, and 4.2TB, respectively. To save space for DiDeMo and ActivitNet datasets, you may extract the frames at a lower resolution by specifying `--frame_size 256`. After the frames are extracted, update the `frames_dir` value in the configuration file `configs/[dataset].json` with the directory of the extracted frames.

If the videos are downloaded via other means, the list of videos may be different from the set used in the paper. In that case, please download the annotation files for the dataset and update the train and test splits. For MSRVTT, we use the annotation files provided by CLIP4Clip (Luo et al., 2021). ActivityNet annotations can be found on the project page of ActivityNet Captions (Krishna et al., 2017). For DiDeMo, we use both the annotations provided by the original author, and the list of videos provided in Collaborative Experts (Liu et al., 2019). Refer to the `README.md` file for specific links to the annotation files. After both the videos and annotation files are downloaded, run `annot_preprocess/[dataset]_preprocess.py` to account for the missing or extra videos. This script requires specifying the directory to the videos and annotation files.

To reproduce the results in the paper, simply execute `scripts/main_results.sh`. Otherwise, you may run the commands individually. More examples can be found in the `README.md` file. You

may need to update the number of GPUs in the file (`--nproc_per_node=num_gpus`) and the batch sizes in the configuration files (`train_batch_size`, `val_batch_size`) accordingly based on your hardware.

After the experiment starts running, it will download all the necessary pretrained models (e.g., CLIP) automatically. The configuration file is mandatory in both the training and evaluation commands. The rest of the arguments are optional and may be different depending on the experimental setting. For example, `--freeze_cnn` freezes the policy backbone, which is the setting used in the paper. To run the no policy setting, we add a flag `--no_policy`. `--diff` and `--concat` use the feature difference and feature concatenation processing methods. Refer to `configs/config.py` for the complete list of arguments and their default values, as well as `main_results.sh` for the main experiments and `ablation_studies.sh` for the ablation studies presented in the paper.

A.6 Evaluation and expected result

After the experiment starts running, the output log will be both printed to the console and saved in `output/log/[datetime]/log.info`. The exact configuration used in the experiment and the best checkpoint can both be found in `output/models/[datetime]`. **After training has finished, the script will automatically use the best checkpoint to perform evaluation on the test set**. Expected evaluation output is as follows:

```
***** Validation information *****
Num examples = 1000
Batch size = 32
Num steps = 32

gflops_table:
clip                : 4.4111 GFLOPS
policy              : 0.3190 GFLOPS
transformer         : 0.0038 GFLOPS
mlp                 : 0.0005 GFLOPS

Num. of queries: 1000, Num. of videos:
1000

CLIP model: 11031 (68.94)%
Skip 1 frames: 4969 (31.06)%
GFLOPS/f: 3.364 GFLOPS/v: 53.832
AVG_FRAMES: 11.031
[t2v_metrics]EVAL epoch 0, R@1: 43.7,
R@5: 71.2, R@10 79.8, R@50 94.5MedR: 2,
MeanR: 15.6
[v2t_metrics]EVAL epoch 0, R@1: 44.4,
R@5: 71.5, R@10 82.1, R@50 95.8MedR: 2,
MeanR: 10.6
```

Note that due to hardware differences, you may see a slight variation in the evaluation results. However, the conclusion of using mmSampler to achieve better performance than the baseline while using fewer frames should still hold.

Technical Appendix

B Alternative Multimodal Retrieval Model

In mmSampler, we finetune the pretrained CLIP ViT-B/32 model (Radford et al., 2021) as the downstream video retrieval model. In this section, we explore a variant of the model, specifically the ViT-B/16 image encoder. ViT-B/16 shares the same architecture as ViT-B/32, but receives smaller patch sizes as input (*i.e.*, more input tokens for the same image size). Since the memory footprint and computational requirements for transformers scale quadratically with input sequence length, a smaller patch size would introduce a more expensive computational cost.

Table 6 illustrates the video-text retrieval performance of the no-policy baseline using ViT-B/16 and mmSampler trained end-to-end with the same model using MobileNetV2 frame features. Compared to the ViT-B/32 baseline, ViT-B/16 shows substantial improvements in all metrics, yet it uses $\sim$4 times more GFLOPs per video. Despite the retrieval benefits the new visual encoder brings, it is not computationally efficient. In general, our design is able to outperform the baseline in almost all the metrics, while saving 35%-51% GFLOPs per video.

In this experiment, we reinforce the fact that mmSampler can be plugged into different video retrieval models to select a subset of salient frames for retrieval, which substantially lowers the inference computational cost.

C Alternative Feature Extractors for the Policy Network

In this section, we explore different feature extractors to encode the frames into vector representations. We consider several lightweight pretrained 2D CNN whose weights are frozen during training, including MobileNetV2 (Sandler et al., 2018), EfficientNet-B2 and EfficientNet-B3 (Tan & Le, 2019). Alternatively, we investigate applying the raw pixels directly instead of forwarding them to a feature-extraction network, which substantially lowers the incurred computational cost. The raw features are obtained by first downsizing the frame into 56×56 pixels, converting it to grayscale, and finally flattening the pixel values into a fixed-size 1D array.

The results for the various feature extractors are presented in Table 7. MobileNetV2 is the cheapest CNN feature extractor out of the three presented, while consistently showing the best performance across all three benchmark datasets. Raw features are the least computational expensive and show marginal improvement over the baseline. On ActivityNet and DiDeMo, raw features are generally inferior to the CNN-based feature extractors in terms of retrieval performance. Efficientnet models have decent performance on the three datasets but they do not display a strong accuracy-efficiency trade-off compared to MobileNetV2.

D GFLOPs Calculation

To estimate the GFLOPs usage for our system, we first break down mmSampler into different components: feature extractor, temporal modelling module, fully connected (FC) layers, and downstream retrieval model. Only the GFLOPs associated with the visual encoder of the CLIP model is computed to compare the savings from selecting a subset of frames. We compute the number of GFLOPs for each component using the `ptflops` library (Sovrasov, 2021), which takes in a PyTorch model and a dummy input to estimate the theoretical multiply-add operations for the model. Number of FLOPs for different components in mmSampler is given in Table 8. Note that for EfficientNet models, we directly take GFLOP numbers reported in their paper.

After obtaining the number of GFLOPs required for each component, we compute the average number of frames used per video during inference for each dataset. The average number of GFLOPs per video is computed by multiplying the number of frames kept by mmSampler with the complexity of the CLIP model, and adding it to the total complexity of the policy network per video. For example, suppose mmSampler used 18 out of 32 frames, the estimated GFLOPs per video is $18 \times 4.4111 + 32 \times (0.3190 + 0.0038 + 0.0005)$ = 89.7 GFLOPS/video. For the no-policy baseline, the number of GFLOPs is simply the total number of uniformly sampled frames multiplied by the GFLOPS of the CLIP visual encoder. In the previous example, the value will be 32×4.4111 = 141.2 GFLOPs/video.

E Qualitative Results

To better understand the sampling behaviour, we also visualize the selected and skipped frames for some videos in Fig. 5. We observe that most of the uninformative or redundant frames are being skipped, which significantly reduces the computational overhead while preserving the key information in the video.

Table 6. Comparison between the no-policy ViT-B/16 baseline denoted as CLIP4Clip and mmSampler for MSRVTT, ActivityNet, and DiDeMo. Frame features processing method is adopted.

			Text $\Longrightarrow$ Video					Video $\Longrightarrow$ Text				
Methods	Datasets	GFLOPs/v	R@1	R@5	R@10	MdR	MnR	R@1	R@5	R@10	MdR	MnR
CLIP4Clip (U, 16)	MSRVTT	281.2	44.5	71.2	80.6	2	14.8	44.0	72.1	81.9	2	11.7
Ours-frame (P, 11.5)	MSRVTT	183.4	45.8	73.1	82.1	2	14.2	45.6	73.6	83.1	2	10.9
CLIP4Clip (U, 32)	ActivityNet	562.5	44.5	75.5	86.6	2	6.4	45.8	75.4	87.2	2	6.2
Ours-frame (P, 16.4)	ActivityNet	297.9	44.2	75.1	86.5	2	6.6	45.9	77.0	87.5	2	6.0
CLIP4Clip (U, 32)	DiDeMo	562.5	42.8	70.7	80.5	2	17.4	42.5	71.7	80.2	2	11.0
Ours-frame (P, 15.2)	DiDeMo	277.3	44.3	72.4	80.9	2	16.4	43.3	71.0	80.1	2	10.4

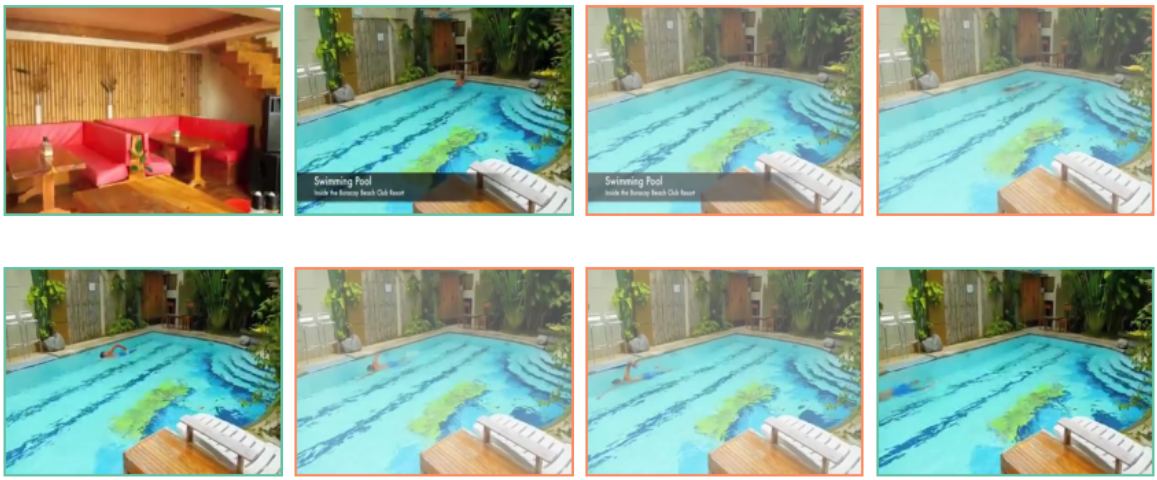
Table 7. Comparison between the no-policy baseline and mmSampler with different feature extractors for MSRVTT, ActivityNet, and DiDeMo. Frame features processing method is adopted.

			Text $\Longrightarrow$ Video					Video $\Longrightarrow$ Text				
Methods	Datasets	GFLOPs/v	R@1	R@5	R@10	MdR	MnR	R@1	R@5	R@10	MdR	MnR
CLIP4Clip (U, 16)	MSRVTT	70.6	42.2	68.7	79.2	2	16.5	42.1	70.4	81.2	2	11.7
Raw (P, 13.6)	MSRVTT	60.1	42.3	69.9	80.1	2	15.6	42.9	70.2	82.1	2	11.3
MobileNetV2 (P, 11.0)	MSRVTT	53.8	43.7	71.2	79.8	2	15.6	44.4	71.5	82.1	2	10.6
EfficientNet-B2 (P, 9.3)	MSRVTT	57.1	42.9	70.2	79.6	2	15.0	42.4	69.8	79.9	2	11.3
EfficientNet-B3 (P, 10.7)	MSRVTT	75.9	41.7	71.1	80.5	2	16.2	42.1	70.4	80.4	2	11.5
CLIP4Clip (U, 32)	ActivityNet	141.2	41.3	72.3	84.0	2	7.5	42.1	74.0	85.1	2	7.0
Raw (P, 15.6)	ActivityNet	69.1	41.6	72.0	83.9	2	7.7	43.3	73.5	85.0	2	7.3
MobileNetV2 (P, 15.9)	ActivityNet	80.4	42.0	72.4	84.1	2	7.4	43.7	74.4	85.8	2	7.0
EfficientNet-B2 (P, 15.4)	ActivityNet	100.3	42.4	72.2	84.1	2	7.6	43.1	74.7	85.2	2	6.9
EfficientNet-B3 (P, 15.9)	ActivityNet	127.8	41.7	72.3	83.8	2	7.8	42.6	73.3	85.3	2	7.3
CLIP4Clip (U, 32)	DiDeMo	141.2	40.7	68.9	79.1	2	18.6	41.0	68.9	79.2	2	12.2
Raw (P, 15.6)	DiDeMo	79.2	40.5	68.7	78.8	2	19.3	41.2	69.7	78.5	2	12.4
MobileNetV2 (P, 19.2)	DiDeMo	94.9	41.4	70.1	80.0	2	18.3	41.8	70.9	80.5	2	11.0
EfficientNet-B2 (P, 15.8)	DiDeMo	102.0	41.4	69.4	79.9	2	18.1	42.0	70.5	79.9	2	11.5
EfficientNet-B3 (P, 14.7)	DiDeMo	122.4	40.9	69.2	78.6	2	18.1	41.6	69.3	79.5	2	11.3

Table 8. GFLOPs look-up table for different components in our system.

Component	GFLOPs
CLIP (ViT-B/32)	4.4111
MobileNetV2	0.3190
1-Layer Transformer	0.0038
FC Layers	0.0005

(a) Four out of eight frames are skipped for video7376. Caption: broth is being added to a soup pot and stirred with a rubber spatula.



(b) Four out of eight frames are skipped for video7797. Caption: a man is swimming in the swimming pool.

(c) Five out of eight frames are skipped for video8661. Caption: a soccer player shoots a goal during a soccer game.

Figure 5. Example outputs for mmSampler on some of the videos in the MSRVTT dataset. Kept frames are outlined in green borders and skipped frames are outlined in red. As in the examples, mmSampler can effectively discard uninformative or redundant frames.