

Scalable Frame Sampling for Video Classification: A Semi-Optimal Policy Approach with Reduced Search Space

Junho Lee¹

joon2003@snu.ac.kr

Jeongwoo Shin¹

swwss@snu.ac.kr

Seung Woo Ko^{1,2}

seungwoo.ko@lgresearch.ai

Seongsu Ha^{1,3}

mars@twelvelabs.io

Joonseok Lee^{*1,4}

joonseok@snu.ac.kr

¹ Seoul National University
Seoul, Korea

² LG AI Research
Seoul, Korea

³ Twelve Labs
Seoul, Korea

⁴ Google Research
Mountain View, CA, USA

*Corresponding author

Abstract

Given a video with T frames, frame sampling is a task to select $N \ll T$ frames, so as to maximize the performance of a fixed video classifier. Not just brute-force search, but most existing methods suffer from its vast search space of $\binom{T}{N}$, especially when N gets large. To address this challenge, we introduce a novel perspective of reducing the search space from $O(T^N)$ to $O(T)$. Instead of exploring the entire $O(T^N)$ space, our proposed semi-optimal policy selects the top N frames based on the independently estimated value of each frame using per-frame confidence, significantly reducing the computational complexity. We verify that our semi-optimal policy can efficiently approximate the optimal policy, particularly under practical settings. Additionally, through extensive experiments on various datasets and model architectures, we demonstrate that learning our semi-optimal policy ensures stable and high performance regardless of the size of N and T .

1 Introduction

As video platforms continue to grow explosively, efficient and scalable video understanding becomes increasingly important. With remarkable advances in deep learning, action recognition and video understanding have also made tremendous progress, from 2D [0, 1, 2, 3, 4, 5, 6] and 3D CNN models [7, 8, 9, 10, 11, 12] to Transformers [13, 14, 15]. Despite these advances, video understanding models are still often overwhelmed by high storage and computational cost. To mitigate this, lighter models with less parameters and computational cost [16, 17, 18, 19, 20] have been proposed.

Focusing mostly on the modeling aspect, however, these works have not touched the most basic underlying condition that digital videos are encoded as a temporal sequence of regularly sampled frames, where each frame is a 2D spatial matrix of regularly sampled pixels, regardless of the content. These regularly sampled pixels are often highly redundant, especially when the frame rate is high. Ideally, it will be more efficient to sample frames proportional to the amount of information, removing redundancy among them as much as possible. Given that most existing video models are still built on a constant frame sampling rate for all videos, there are further room for improvement in terms of efficiency.

In this paper, we consider the frame sampling problem. For a video with T frames, we propose a frame sampler that selects $N \ll T$ frames, which are then used by a video model pre-trained for a specific downstream task (e.g., classification). The goal of this task is to train a sampler that finds the best combination of N frames out of the given T frames, letting the pre-trained model (classifier) to achieve the highest downstream task performance.

To achieve this, the sampler learns a *policy* to select N frames out of T candidates. The ideal policy would select the best N frames that lead to the highest downstream task performance, and we call this *optimal policy* (π_o). A naive approach would collect the best combinations as training data and train the sampler in a supervised manner. However, finding the best combination is an NP-hard combinatorial optimization problem, as it requires comparing all $\binom{T}{N}$ possible combinations to find the solution. Exhaustively searching this $O(T^N)$ space is practically infeasible when N and T grow.

Several previous works [22, 24, 41, 42, 44] have applied reinforcement learning to search this space, setting up the sampler as an agent and the classifier as the environment and training the sampler with a reward function designed to find π_o . While they have shown promising results for small N and T , the underlying search space of $O(T^N)$ still remains the same, posing challenges for large N and T .

Considering the purpose of the frame sampling task, however, it is more important for the sampler to operate effectively on a long video with a large N and T . Unlike the previous works that operate directly within the $O(T^N)$ space, we take a step back to find a way to reduce this search space itself. The fundamental reason for this exponential growth of search space is that the value of a selected frame depends on those of other selected frames and thus we need to consider the joint distribution of the frame values.

What if, however, the value of a frame can be fairly determined independently from each other? We would be able to score T frames independently and simply to take the top N frames. In this paper, we argue that, based on experimental results, frames can be reasonably assessed independently under most practical conditions where frame sampling is relevant—namely, when the video is not excessively short and the frame rate is not extremely high. We observe that a policy based on evaluating the value of each frame can reasonably approximate an optimal sampling policy.

Based on this observation, we propose a *semi-optimal policy* (π_s) that selects the top N frames based on estimated value of each frame using per-frame confidence. This approach significantly reduces the search space to $O(T)$. We empirically show on multiple datasets that π_s is a policy that reasonably approximates π_o , and demonstrate that the state-of-the-art sampler [24] achieves more stable and superior performance when it learns π_s instead of π_o , not only with small but also large N and T .

Our contributions are summarized as follows:

- We introduce a novel perspective of *reducing the search space from $O(T^N)$ to $O(T)$* in frame sampling for video classification, proposing the semi-optimal policy.

- Through various analyses, we demonstrate that the semi-optimal policy approximates the optimal policy *under the most practical settings*.
- From extensive experiments, we show that a sampler learning the semi-optimal policy achieves stable and high performance across both small and large values of N and T .

2 Related Work

Video Recognition. 2D and 3D convolutional neural networks (CNNs) have been widely used for image and video classification. 2D-CNN encodes each frame individually, then aggregates the temporal dynamics by feature pooling [13, 21, 45], rank pooling [9], or relation network [46]. Two-stream methods take both spatial stream from CNN and temporal stream from optical flow as input at various levels of fusion [4, 42]. 3D-CNN approaches apply convolution operations on both spatial and temporal dimensions simultaneously, pioneered by C3D [45]. I3D [9] combines the 3D convolution with two-stream approaches. P3D [29] and R(2+1)D [46] factorizes the spatial and temporal dimensions in the 3D convolution. SlowFast [8] model consists of two different pathways of focusing more on temporal and spatial information, respectively. CSN [32] shows that separating the channel interaction with spatio-temporal interaction leads to lower computational cost and better regularization.

Recently, Transformers [38] have been applied to video classification and action recognition. Due to the high computational cost for processing all patches in the video, most video Transformer models factorize spatial and temporal attentions [10, 1] or utilize the inductive bias of locality in videos [25] for efficient handling of videos.

Efficient Video Recognition. One of the biggest challenges of video classification is the computational cost stemming from the vast number of frames. One remedy is to lighten the model architecture itself [6, 9, 16, 28, 39, 47]. Another direction is algorithmic improvement for better efficiency. AdaFrame [44] learns from both the current frame and the global context to make decisions on where to look next in the video. AdaFocus series [40, 41] use a lightweight global CNN to guide the policy network to extract the most useful image crop to classify the video. On-the-fly gating techniques are introduced to reduce computation by only extracting finer features at selected frames [43], determining an early exiting point [10], or deciding which frames in the video to fully process [30]. ListenToLook [10] utilizes only a single frame and audio stream to determine the full video descriptor for the clip.

An alternative is to sample frames to process, instead of feeding all of them to the classification model, vastly reducing the computational cost. ARNet [26] and VideoIQ [63] pre-train multiple classifiers and train a sampler to determine which classifier to process each frame. MARL [42] utilizes multiple agents that adjust the sampling location based on local and historical context. OCSampler [24] simply selects the salient frames, while Ada2D [22] determines which frames will be used in 2D or 3D. Ours falls into this category, particularly aiming to effectively sample N frames for a large T .

3 Problem Formulation

Sampling Scenario. We assume an offline environment, where we are given T candidate frames $\mathbf{v} \in \mathbb{R}^{T \times 3 \times H \times W}$ from a video, with H and W indicating the height and width of the frames, respectively. A frozen pretrained classifier $f_c: \mathbb{R}^{N \times 3 \times H \times W} \rightarrow [0, 1]^C$ is given, where N is the number of input frames and C is the number of classes. In this context, our final

objective is to train a sampler that finds the optimal N frames out of the given T frames that make the classifier to best perform on the downstream task.

Optimal Policy π_o . Given T candidate frames, the optimal policy π_o chooses N frames that maximize the confidence of the classifier f_c on the true label $y \in [1, \dots, C]$, among all possible combinations of the candidate frames. We define the frames selected by π_o as optimal set.

Challenges. The most straightforward approach to learn π_o would be to train a sampler in a supervised manner to select the optimal set of frames. However, finding the optimal set of frames is computationally prohibitive, requiring to explore a search space of $O(T^N)$. Even for moderately large N and T , the search space quickly grows, making it challenging to train the sampler directly.

To explore this space, previous works adopt reinforcement learning, where a network sampler (agent) is trained based on rewards given by the classifier (environment) for the selected frames (action) according to the policy. Although these approaches have been effective for small N and T , they are not scalable for growing N and T . To mitigate the exponential time complexity caused by this exponentially growing search space, we propose an effective approach in most practical settings.

4 Method

4.1 Semi-Optimal Policy

The main challenge of the frame sampling problem lies in the fact that the importance of a frame is influenced by other selected frames, necessitating to explore the interactions in the exponentially growing search space to N and T . Conversely, we recognize that the problem would be much simpler if: 1) the importance of each frame were (roughly) independent from other frames, and 2) we could reasonably approximate the importance of each frame. This would allow us to score T frames individually and select the top N of them.

In this section, we show that the frames are indeed close to independence when it comes to a practical setting of the frame sampling task – where the video is not trivially short and the frame rate is relatively low. The opposite case, a short snippet with high frame rate, would not need a sophisticated frame sampling method anyway, since most frames would be highly redundant. From this observation, we introduce a new sampling policy that effectively approximates the optimal policy while significantly reducing the space complexity.

Independence between Frames. We claim that frames influence each other’s importance mainly due to the overlapping information between them. That is, even if a frame contains important information, its importance will be diminished if the same information is redundantly provided by others. Conversely, the importance of a frame increases if it contains accurate and unique information; in other words, if it provides distinct information that is not covered by other frames. Thus, if a candidate frame is sufficiently dissimilar from others, we would be able to independently score it without concerning redundancy.

To check applicability of this idea, we measure the similarity between adjacent frames at various frame rates. The similarity can be computed in several resolutions; as coarse as the probability distribution over the class labels, or as fine-grained as the pixel-level. Both extremes are less ideal, since they do not convey high-level semantics of each frame. Thus, we estimate the relevance between two sampled frames i and j by applying a Gaussian

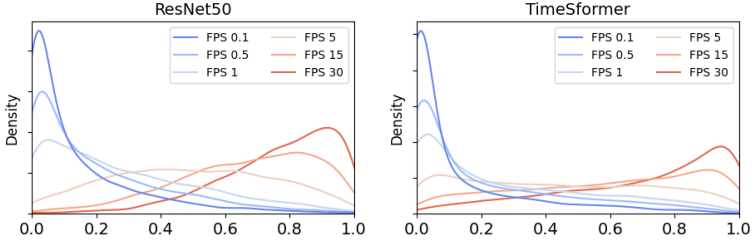


Figure 1: **Distribution of $\mathcal{I}(\mathbf{x}_i, \mathbf{x}_j)$ with kernel density estimate.**

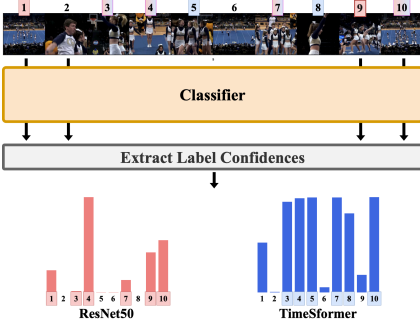


Figure 2: **Illustration of the Semi-Optimal Policy π_s .** We briefly illustrate how π_s works on two different architecture. The numbers in the pink and blue boxes represent the frame indices sampled with ResNet50 and TimeSformer as the backbone, respectively, when $N = 6$.

smoothing kernel to their visual embeddings, denoted by \mathbf{x}_i and \mathbf{x}_j :

$$\mathcal{I}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{Z} e^{\{-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)\}}, \quad (1)$$

where Σ determines the kernel bandwidth and $Z = 1/|\sqrt{2\pi\Sigma}|$ is the normalizer.

Fig. 1 illustrates the estimated relevance between two consecutive frames at various frame rates on ActivityNet-v1.3, using ResNet50 and TimeSformer features. We observe with both features that higher frame rates lead to higher redundancy between frames. Although this tendency is expected considering the nature of videos, what we need to focus is *where is the proper frame rate threshold* that we can safely assume independence between frames. From the plots, we conclude that we can reliably assume independence up to 1 fps, while 5 fps is on the borderline. Under a practical setting for frame sampling, we claim that 1 fps is still reasonable, since higher frame rates would be prohibitively expensive for long videos; *e.g.*, for a 5-min-long video, 5 fps already piles up 1,500 candidate frames. In short, we can reasonably assume independence among frames when the target video is not trivially short, since we would only feed regularly sampled frames at lower frame rate due to the computational overhead.

Semi-optimal Policy π_s . Under the frame independence condition, we score the importance of each frame independently. Specifically, we infer the frame-level confidence scores for all C classes by treating each frame $\mathbf{v}_t \in \mathbb{R}^{H \times W \times 3}$ as a single-frame video clip $\mathbf{v}_t \in \mathbb{R}^{1 \times H \times W \times 3}$ and take its confidence scores $f_c(\mathbf{v}_t) \in [0, 1]^C$. We illustrate the mechanism of π_s in Fig. 2.

In order to score the importance of a frame, its confidence scores $f_c(\mathbf{v}_t)$ on C classes need be aggregated into a single confidence score $c(\mathbf{v}_t) \in \mathbb{R}$. There are two options for this. First, we may take the confidence for the true label y as importance score. That is, $c(\mathbf{v}_t) = [f_c(\mathbf{v}_t)]_y$, where $[f_c(\mathbf{v}_t)]_y$ indicates the confidence of frame \mathbf{v}_t for the label y using the classifier f_c . This approach aims to learn the desired importance score for the actual

f_c	Policy	A-Net (mAP)	M-Kin. (Top-1)
TimeSformer	π_u	87.0%	79.6%
	π_o	91.5%	89.3%
	π_s	89.4%	84.8%
	All	89.0%	81.2%
ResNet50	π_u	75.3%	72.5%
	π_o	90.5%	83.8%
	π_s	87.4%	80.3%
	All	77.8%	73.6%

Table 1: **Performance of π_o and π_s on ActivityNet and Mini-Kinetics.** Relative improvement from π_u is provided on the right.

Dataset	Sampler	Sampling Fidelity (%)					
		$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$
A-Net	Random	10.0	20.0	30.0	40.0	50.0	60.0
	FrameExit	10.2	19.3	29.3	39.3	49.6	59.3
	π_s	100.0	74.6	73.2	75.1	78.5	81.0
M-Kin.	Random	10.0	20.0	30.0	40.0	50.0	60.0
	FrameExit	9.8	22.5	32.0	42.5	51.5	62.5
	π_s	100.0	61.5	65.2	70.6	71.8	80.5

Table 2: **Sampling Fidelity.** Note that we report the expected value of the sampling fidelity for random sampling.

class. The second option takes the maximal confidence across all classes $i = 1, \dots, C$; that is, $c(v_t) = \max_{i=1, \dots, C} [f_c(v_t)]_i$. Motivation of this approach is to train the sampler with more consistent scores, since the importance may significantly vary depending on the labels even for similar scenes. Once we have the frame-wise confidence distribution $c(v_t)$ for $t = 1, \dots, T$, we select the top N frames with highest importance score. That is,

$$\pi_s(v, y; N) = \text{top-}N \ c(v_t)_{t=1, \dots, T}. \quad (2)$$

Empirical Verification. We verify that our proposed policy π_s reasonably approximates the optimal policy π_o in practice. First, we actually compare the frame sampling performance of π_s against π_o with two architectures, a CNN and a transformer, on two datasets, ActivityNet-v1.3 [14] and Mini-Kinetics [15], representing the untrimmed and trimmed video datasets, respectively. Tab. 1 compares the performance of a pre-trained classifier when it takes a set of $N = 6$ frames selected from $T = 10$ frames by various sampling policies: uniform sampling (π_u), the optimal policy (π_o), our semi-optimal policy (π_s), and using all frames without sampling (All). The results indicate that π_s demonstrates significant improvement over π_u and ‘All’ under all settings, most closely approximating the optimal policy, π_o .

Additionally, we report the sampling fidelity, which indicates the similarity of the sampled set to the optimal set, across various values of N . Formally, sampling fidelity is defined as $\mathbb{E}_v (|\mathcal{S}_t(v) \cap \mathcal{S}_o(v)|/N)$, where $\mathcal{S}_t(v)$ and $\mathcal{S}_o(v)$ denote the sampled set of frames from the video v by the target policy (π_t) and π_o , respectively. Tab. 2 compares sampling fidelity of our π_s against two baselines: the random sampling and a deterministic coarse-to-fine sampling policy proposed in FrameExit [16]. We observe that our π_s demonstrates significantly higher sampling fidelity than the FrameExit policy, which exhibits fidelity nearly identical to the random policy. We also provide a qualitative comparison in Appendix C.1–C.2.

4.2 SOSampler: Semi-optimal Policy-based Sampler

As shown in Sec. 4.1, our semi-optimal policy π_s reasonably approximates the optimal policy π_o . To train a lightweight network sampler based on this policy, we propose the Semi-Optimal Sampler (SOSampler), which learns π_s instead of π_o in a straightforward manner.

Overview. Fig. 3 illustrates an overview of our approach, which consists of a frame sampler \mathcal{S}_{SO} and an action classifier f_c . At training, we first take T candidate frames from the input video and compose a video clip v . The classifier f_c takes each candidate frame as a single-frame clip input and conducts classification. This process is performed individually for each

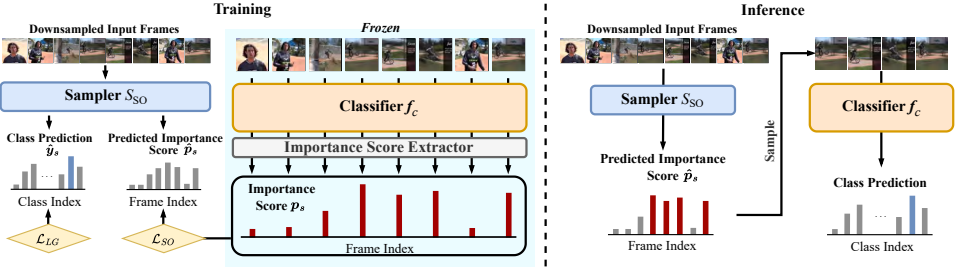


Figure 3: **SOSampler Algorithm.** SOSampler consists of a sampler S_{SO} and a classifier f_c , which can be any model architecture.

frame, producing T predictions in total. From these T predictions, we take the representative confidence $c(v_t)$ either by taking the true label y or by taking the maximal one, as explained in Sec. 4.1. We then apply softmax to get a normalized importance distribution $\mathbf{p}_s \in \mathbb{R}^T$. Then, our sampler S_{SO} takes the *spatially down-sampled* version of the same T candidate frames. S_{SO} predicts the importance scores $\hat{\mathbf{p}}_s \in \mathbb{R}^T$ over them and classifies each frame into the C classes using linear projections. See Appendix A.1 for more detailed description.

Training Objectives. We train the sampler S_{SO} with two objectives: 1) the semi-optimal policy loss \mathcal{L}_{SO} , penalizing when the estimated importance score $\hat{\mathbf{p}}_s$ does not agree with the confidence \mathbf{p}_s produced by f_c , and 2) the label guidance loss \mathcal{L}_{LG} , penalizing when the predicted class of each frame differs from the true label to ensure that each frame contains information about the video-level class.

For \mathcal{L}_{SO} , we initially experimented with mean square loss, $\|\hat{\mathbf{p}}_s - \mathbf{p}_s\|_2$, but the result was not satisfactory. Taking inspiration that what we need is a correct ordering of the importance scores, not their exact values, we adopt a pairwise ranking loss [19] for \mathcal{L}_{SO} , which penalizes the model when the relative order of importance between a pair of two frames is reversed:

$$\mathcal{L}_{SO} = \sum_{(p_i, p_j) \in \Psi} \text{sign}(p_i - p_j) \cdot \max(\gamma + \hat{p}_i - \hat{p}_j, 0), \quad (3)$$

where $\Psi = \{(p_i, p_j); p_i > p_j\}$, $\text{sign}(z)$ is the sign function, and γ is a hyperparameter indicating a target margin. For \mathcal{L}_{LG} , we use the cross-entropy loss.

The overall loss function is constructed by

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{SO}(\mathbf{p}_s, \hat{\mathbf{p}}_s) + (1 - \lambda) \cdot \mathcal{L}_{LG}(\hat{\mathbf{y}}_s, \mathbf{y}), \quad (4)$$

where λ is a hyperparameter that adjusts the ratio of the two losses, $\hat{\mathbf{y}}_s$ denotes the predicted confidence scores by the sampler and \mathbf{y} is the one-hot encoding of the true label y .

Inference. At inference, T candidate frames are uniformly sampled from the target video. As in training, each frame is down-sampled and fed into the sampler S_{SO} to obtain the importance scores. Frames with the top N scores are selected, and those selected frames in the original resolution are fed into f_c to perform the final classification.

Methods	Back-bones	ActivityNet		Mini-Kinetics	
		mAP	GFLOPs	Top-1	GFLOPs
LiteEval [14]	Res-Net50	72.7%	95.1	61.0%	99.0
SCSampler [14]		72.9%	42.0	70.8%	41.9
AR-Net [24]		73.8%	33.5	71.7%	32.0
videoIQ [14]		74.8%	28.1	72.3%	20.4
AdaFocus [14]		75.0%	26.6	72.9%	38.6
FrameExit [14]		76.1%	26.1	72.8%	19.7
OCSampler [14]		77.2%	25.8	73.0%	21.6
SOSampler		77.7%	25.8	73.5%	21.6
Ada2D [14]	Slow	84.0%	70.1	79.2%	738
OCSampler [14]	Only	87.3%	68.2	82.6%	27.3
SOSampler	50	88.0%	64.0	83.0%	27.3
FrameExit [14]	X3D-S	86.0%	9.8	—	—
OCSampler [14]		86.6%	7.9	—	—
SOSampler		87.2%	7.6	—	—
OCSampler [14]	TimeS-former	83.2%	76.8	80.7%	76.8
SOSampler		88.7%	76.8	80.7%	76.8

Table 3: **Comparison on ActivityNet-v1.3 and Mini-Kinetics for small N and T .** The best performing model is **bold-faced**.

Methods	Back-bones	Mini-Sports1M		COIN	
		mAP	GFLOPs	Top-1	GFLOPs
LiteEval [14]	Res-Net50	44.7%	66.2	—	—
SCSampler [14]		44.3%	42.0	79.8%	42.0
AR-Net [24]		45.0%	37.6	—	—
AdaFuse [14]		44.1%	60.3	—	—
OCSampler [14]		46.7%	25.8	80.1%	25.8
SOSampler		48.3%	25.8	80.7%	25.8
OCSampler [14]	TimeS-former	45.6%	76.8	81.4%	76.8
SOSampler		49.1%	76.8	87.7%	76.8

Table 4: **Comparison on Mini-Sports1M and COIN Comparison on for small N and T .** The best performing model is **bold-faced**.

5 Experiment

5.1 Experimental Setup

Datasets. We evaluate on four public video classification benchmarks: ActivityNet-v1.3 [9], Mini-Kinetics [14], Mini-Sports1M [14], and COIN [34]. See Appendix A.2 for more detailed description about the datasets.

Experimental Protocol. We randomly sample T frames from each video as a training example and uniformly sample T frames from each test video. We use MobileNetv2TSM [23, 30] as the feature extractor of our sampler S_{SO} and one additional fully-connected layer at the head of the sampler, following [24]. See Appendix A.3 for more details.

Evaluation Metrics. Following the common practice, we measure the mean average precision (mAP), the mean of class average precisions, for ActivityNet, Mini-Sport1M, and COIN datasets. For Mini-Kinetics, we use top-1 accuracy, which is the ratio of the correctly classified test samples. For computational cost, we report GFLOPs for all datasets.

5.2 Results and Analysis

Comparison with Small N and T . To compare with existing methods under the same conditions,¹ we first evaluate with the common setting of $T = 10$ and $N = 6$ (except for Mini-Kinetics with ResNet50, where $N = 5$).

As seen in Tab. 3–4, our approach consistently outperforms previous methods across all four datasets and backbone architectures, highlighting the effectiveness of learning π_s even when the search space is not very large. Nevertheless, comparing Tab. 3 to Tab. 1, we observe that SOSampler still does not fully emulate π_s , possibly due to the loss when it transfers the label information. A future work may further address this issue to fill the gap.

¹Results for OCSampler with ResNet50 in Tab. 3 is our reproduction using pre-trained weights and settings provided by the original author. They slightly lag behind on Mini-Kinetics, by 0.7%.

Dataset	Backbone	Method	N / T		
			8 / 30	16 / 60	32 / 100
ActivityNet	ResNet50	Uniform	77.1%	79.4%	80.4%
		OCSampler	78.0%	79.1%	80.1%
		SOSampler	78.7%	80.2%	81.1%
	TimeSformer	Uniform	88.5%	89.9%	90.3%
		OCSampler	85.0%	85.3%	84.5%
		SOSampler	89.5%	90.1%	90.5%
Mini-Sports1M	ResNet50	Uniform	46.9%	48.8%	49.1%
		OCSampler	48.6%	49.6%	50.0%
		SOSampler	50.0%	51.1%	51.5%
	TimeSformer	Uniform	53.9%	55.6%	56.8%
		OCSampler	48.9%	49.6%	50.0%
		SOSampler	55.1%	56.9%	57.8%

Table 5: **Experiment on long videos for large N and T .** The best performing model is **bold-faced**.

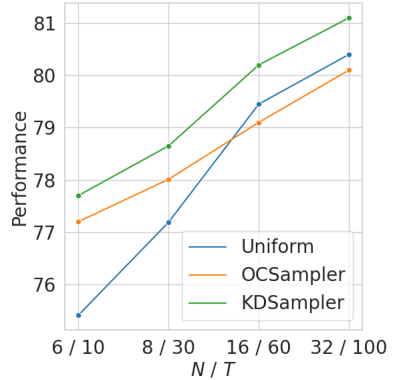


Figure 4: **Performance by sampler across N/T on ActivityNet-v1.3.**

Comparison with Large N and T . We conduct experiments on large N and T to verify the effect of reduced search space by our proposed policy π_s , comparing with OCSampler [24] and uniform sampling as baselines. Note that OCSampler and SOSampler have exactly the same architecture – the only difference lies in the learning objective. We evaluate on the ActivityNet-v1.3 and Mini-Sports1M datasets, which consist of sufficiently long videos², to ensure the frame independence even for a large T .

As shown in Tab. 5, our approach consistently achieves competitive performance regardless of the backbone architectures and with various values of N and T . With the OCSampler, on the other hand, the performance improvement is inconsistent and it sometimes underperforms even than the uniform sampling. This is illustrated in Fig. 4 as well; the performance of OCSampler lags behind the uniform sampling from $N = 16, T = 60$ and beyond, indicating that directly searching the $O(T^N)$ space is not scalable for large N and T . Our semi-optimal policy does not suffer from this, recalling that our approach adopts the same architecture as OCSampler. Additionally, we observe that OCSampler does not perform well with the TimeSformer backbone. We also attribute this to the more complex search space of transformer-based classifiers compared to CNN-based ones. Based on the results, we conclude that our approach to reduce the complexity of search space is indeed effective for large N and T , which has been challenging for existing methods.

Qualitative Analysis. In Appendix C.1–C.2, we illustrate success and failure cases of our SOSampler, comparing with sampling policies of π_o and π_s .

5.3 Ablation Study

Loss function. Tab. 6 reports the result of an ablation study on the loss functions, conducted on ActivityNet-v1.3. ‘MSE’ and ‘Ranking’ indicate the two choices for \mathcal{L}_{SO} presented in Sec. 4.2, while ‘Label’ and ‘Max’ indicate how we obtain the importance score of each frame $c(v_t)$ from the confidence distribution $f_c(v_t)$ over all classes.

As we mentioned earlier, the ranking loss in Eq. (3) outperforms the MSE loss for \mathcal{L}_{SO} . We also observe that the max-aggregation leads to stronger performance than the label confidence, probably because it learns consistent scores across features. The label guidance loss

²Results for short videos are discussed in Appendix B.1.

\mathcal{L}_{SO}				\mathcal{L}_{LG}	mAP (%)
MSE	Ranking	Label	Max		
✓		✓			87.38
✓		✓		✓	87.51
	✓	✓			88.21
	✓	✓		✓	88.40
	✓		✓	✓	88.65

Table 6: **Ablation study on losses.**

T	10				6	10	16	24
N	2	4	6	8	6			
mAP	71.1%	76.2%	77.7%	77.9%	75.3%	77.7%	78.2%	78.4%
GFLOPs	9.3	17.6	25.8	34.1	24.7	25.8	26.4	27.2

Table 7: **Ablation study on T and N .**

\mathcal{L}_{LG} turns out to slightly improve the overall performance.

Exploration on N and T . We fix $N = 6$ and $T = 10$ in Tab. 3–4 mainly for the purpose of comparison with previous models. We further explore various combinations of $N \in \{2, 4, 6, 8\}$ and $T \in \{6, 10, 16, 24\}$ to better understand our method. As seen in Tab. 7, performance consistently improves with a larger N , but the gain diminishes beyond $N \geq 6$. Also in the case of $N = 6$, we observe consistently improved performance with larger T .

6 Conclusion

In this paper, we address the scalability challenge of the frame sampling task, proposing our novel semi-optimal policy π_s to dramatically reduce the search space itself from $O(T^N)$ to $O(T)$, supported by empirical evidence of frame independence. Through extensive experiments, we demonstrate that π_s effectively approximates the optimal policy. Furthermore, across all datasets and architectures, the sampler which learns π_s instead of π_o outperforms existing methods both for small and large N and T . These results suggest that our new approach, which changes the search space itself rather than the exploration method, is more effective than previous approaches. However, the frame independence assumption we propose does not hold in all scenarios, and there remains potential for further performance improvements in the sampler learning our proposed π_s . Addressing these limitations will be an important direction for future work.

Acknowledgements

This work was supported by the New Faculty Startup Fund from Seoul National University, by Samsung Electronics Co., Ltd (IO230414-05943-01, RAJ0123ZZ-80SD), by Youlchon Foundation (Nongshim Corp.), and by National Research Foundation (NRF) grants (No. 2021H1D3A2A03038607/50%, RS-2024-00336576/10%, RS- 2023-00222663/5%) and Institute for Information & communication Technology Planning & evaluation (IITP) grants (No. RS-2024-00353131/25%, RS-2022-II220264/10%), funded by the government of Korea.

References

- [1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *ICCV*, 2021.
- [2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you

- need for video understanding? In *Proc. of the International Conference on Machine Learning (ICML)*, 2021.
- [3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.
 - [4] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Nieves. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.
 - [5] Quanfu Fan, Chun-Fu Richard Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depth-wise temporal aggregation. *NeurIPS*, 32, 2019.
 - [6] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, 2020.
 - [7] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.
 - [8] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *ICCV*, 2019.
 - [9] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):773–787, 2016.
 - [10] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. In *CVPR*, 2020.
 - [11] Amir Ghodrati, Babak Ehteshami Bejnordi, and Amirhossein Habibian. FrameExit: Conditional early exiting for efficient video recognition. In *CVPR*, 2021.
 - [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
 - [13] Seong Jae Hwang, Joonseok Lee, Balakrishnan Varadarajan, Ariel Gordon, Zheng Xu, and Paul Natsev. Large-scale training framework for video annotation. In *Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining*, 2019.
 - [14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.
 - [15] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The Kinetics human action video dataset. *arXiv:1705.06950*, 2017.
 - [16] Okan Kopuklu, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource efficient 3D convolutional neural networks. In *ICCV*, 2019.
 - [17] Bruno Korbar, Du Tran, and Lorenzo Torresani. SCsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

- [18] Hyodong Lee, Joonseok Lee, Joe Ng, and Paul Natsev. Large scale video representation learning via relational graph clustering. In *CVPR*, 2020.
- [19] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *Proc. of the ACM International Conference on World Wide Web (WWW)*, 2014.
- [20] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Paul Natsev. Collaborative deep metric learning for video understanding. In *Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining*, 2018.
- [21] Joonseok Lee, Paul Natsev, Walter Reade, Rahul Sukthankar, and George Toderici. The 2nd youtube-8m large-scale video understanding challenge. In *ECCV*, 2018.
- [22] Hengduo Li, Zuxuan Wu, Abhinav Shrivastava, and Larry S Davis. 2D or not 2D? adaptive 3D convolution selection for efficient video recognition. In *CVPR*, 2021.
- [23] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *ICCV*, 2019.
- [24] Jintao Lin, Haodong Duan, Kai Chen, Dahua Lin, and Limin Wang. OCSampler: Compressing videos to one clip with single-step sampling. In *CVPR*, 2022.
- [25] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *CVPR*, 2022.
- [26] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. AR-net: Adaptive frame resolution for efficient action recognition. In *ECCV*, 2020.
- [27] Yue Meng, Rameswar Panda, Chung-Ching Lin, Prasanna Sattigeri, Leonid Karlinsky, Kate Saenko, Aude Oliva, and Rogerio Feris. AdaFuse: Adaptive temporal fusion network for efficient action recognition. *arXiv:2102.05775*, 2021.
- [28] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks. *Applied AI Letters*, 3(1):e38, 2022.
- [29] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3D residual networks. In *ICCV*, 2017.
- [30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.
- [31] Jonghyeon Seon, Jaedong Hwang, Jonghwan Mun, and Bohyung Han. Stop or forward: Dynamic layer skipping for efficient action recognition. In *WACV*, 2023.
- [32] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *NIPS*, 27, 2014.
- [33] Ximeng Sun, Rameswar Panda, Chun-Fu Richard Chen, Aude Oliva, Rogerio Feris, and Kate Saenko. Dynamic network quantization for efficient video inference. In *ICCV*, 2021.

- [34] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. COIN: A large-scale dataset for comprehensive instructional video analysis. In *CVPR*, 2019.
- [35] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.
- [36] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.
- [37] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.
- [38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS*, 30, 2017.
- [39] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *CVPR*, 2020.
- [40] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. In *ICCV*, 2021.
- [41] Yulin Wang, Yang Yue, Yuanze Lin, Haojun Jiang, Zihang Lai, Victor Kulikov, Nikita Orlov, Humphrey Shi, and Gao Huang. AdaFocus V2: End-to-end training of spatial dynamic networks for video recognition. In *CVPR*, 2022.
- [42] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, and Shilei Wen. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *ICCV*, 2019.
- [43] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. LiteEval: A coarse-to-fine framework for resource efficient video recognition. *NeurIPS*, 32, 2019.
- [44] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. AdaFrame: Adaptive frame selection for fast video recognition. In *CVPR*, 2019.
- [45] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.
- [46] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.
- [47] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. ECO: Efficient convolutional network for online video understanding. In *ECCV*, 2018.

Appendix

A Detailed Descriptions

A.1 Model Configuration

Sampler. The sampler S_{SO} takes a sequence of T frames as input, composed of a lightweight feature extractor f_s , an importance predictor h_s , and an action classifier h_c .

Given T candidate frames $\mathbf{v} \in \mathbb{R}^{T \times 3 \times H \times W}$, our sampler first spatially downsamples them to $\mathbf{v}' \in \mathbb{R}^{T \times 3 \times H' \times W'}$, where $W' < W$ and $H' < H$. Then, a 2D image representation network $f_s : \mathbb{R}^{3 \times H' \times W'} \rightarrow \mathbb{R}^D$ extracts frame-level features, where D denotes the dimensionality of the feature. Inferring all T frames using f_s , a feature map

$$\mathbf{z} = \{f_s(\mathbf{v}'_1), \dots, f_s(\mathbf{v}'_T)\} \in \mathbb{R}^{T \times D} \quad (5)$$

is constructed, where \mathbf{v}'_i denotes the i -th frame of \mathbf{v}' .

Then, our sampler conducts two downstream tasks using the extracted features \mathbf{z} . First, it estimates the frame importance score \mathbf{p}_s using a regressor $h_s : \mathbb{R}^{T \times D} \rightarrow \mathbb{R}^T$:

$$\hat{\mathbf{p}}_s = \{h_s(\mathbf{z}_1), \dots, h_s(\mathbf{z}_T)\}. \quad (6)$$

Second, it performs the downstream classification task. Using a frame-level classifier $h_c : \mathbb{R}^{T \times D} \rightarrow \mathbb{R}^C$, it predicts the relevance of each frame $t = 1, \dots, T$ for the C classes, and these predictions are aggregated over the T frames to a video-level prediction by taking the average:

$$\hat{\mathbf{y}}_s = \frac{1}{T} (h_c(\mathbf{z}_1) + \dots + h_c(\mathbf{z}_T)). \quad (7)$$

Note that h_c is used only during training to make the backbone f_s learn the label information. We simply implement h_s and h_c with linear projections followed by a softmax.

Classifier. The classifier f_c can be any visual recognition model, such as a 2D or 3D CNN, or a Transformer, pretrained and frozen throughout the training. During training, f_c is used to compute the importance score, which serves as a pseudo-label to train the sampler S_{SO} by distilling the knowledge from the classifier f_c .

During inference, f_c is used to perform the downstream task on a clip $\mathbf{v}^s \in \mathbb{R}^{N \times 3 \times H \times W}$ of sampled N frames:

$$\hat{\mathbf{y}} = f_c(\mathbf{v}^s). \quad (8)$$

Our goal is to train the sampler S_{SO} so that the frozen classifier f_c predicts $\hat{\mathbf{y}}$ close to the ground truth label \mathbf{y} , the one-hot encoding of the true label \mathbf{y} .

A.2 Dataset

ActivityNet-v1.3 includes 10,024 training and 4,926 validation videos. The average video length is 117 seconds with an average of 3,335 frames, covering 200 categories. Mini-Kinetics, a subset of Kinetics400, comprises 121,215 training and 9,867 validation videos. The videos have an average duration of 10 seconds with an average of 261 frames, covering 200 categories. Mini-Sports1M, a subset of Sports1M [14], consists of 14,586 training and 4,855 validation videos. The average video length is 330 seconds with an average of 4,467 frames, covering 487 action classes. COIN is composed of 11,827 YouTube videos related to 180 different tasks. The videos have an average length of 141 seconds with an average of 4,009 frames.

Dataset	Backbone	Method	N / T		
			8 / 30	16 / 60	32 / 100
Mini-Kinetics	ResNet	OCSampler	73.52%	74.00%	74.17%
		SOSampler	73.79%	74.46%	74.65%
	TimeSformer	OCSampler	79.13%	78.05%	76.33%
		SOSampler	79.93%	80.44%	81.32%
COIN	ResNet	OCSampler	78.69%	79.79%	80.06%
		SOSampler	79.13%	80.21%	81.02%
	TimeSformer	OCSampler	80.88%	80.90%	81.20%
		SOSampler	86.52%	87.37%	88.08%

Table I: **Experiment on short video datasets for large N and T .** The best performing model is **bold-faced**.

A.3 Implementation Details

We follow the preprocessing steps outlined in [24]. We sample T frames from each video as a training example. All frames are randomly scaled and cropped to 224×224 , followed by random flipping for augmentation. We then reduce the resolution of each frame to 128×128 before feeding them into our sampler S_{SO} . During inference, we uniformly sample T frames from a test video, resize them to 128×128 , and feed them to the sampler S_{SO} . Then, we feed the original 224×224 images of the selected frames to f_c .

For the pretrained classifier weights, we utilize the pretrained weights provided by [14] on the ActivityNet-v1.3 and Mini-Kinetics datasets with the ResNet50 classifier. For other datasets and architectures, we train the classifier from scratch.

To train our SOSampler, we use a learning rate of 10^{-3} and set $\lambda = 0.99$ for all datasets. We optimize our loss function in Eq. (4) using the stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and weight decay set to 10^{-4} . We employ cosine annealing as a learning rate scheduler without warm-up.

We implement our method using PyTorch and train on a single NVIDIA A100 GPU with 40GB of memory.

B Additional Results

B.1 Comparison with Large N and T on Short Videos

In short videos, as T increases, the FPS becomes significantly higher, weakening our assumption of independence between frames. Therefore, our approach does not sufficiently improve the performance as N and T increase. However, it consistently shows an upward trend and still outperforms OCSampler in all settings. This result suggests that our method is still superior to the existing method, even in the FPS range where our assumption of independence between frames is weak.

B.2 Computational Efficiency

While GFLOPs serve as a metric for measuring the efficiency of a model, it does not provide the actual running time. Therefore, we additionally compare the actual inference

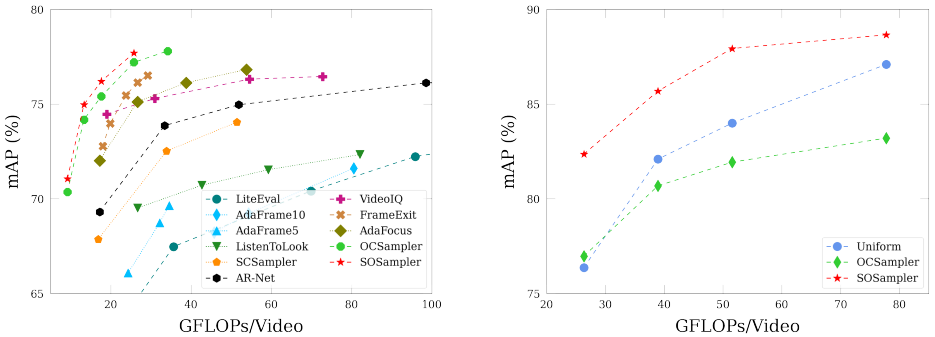


Figure I: **Mean Average Precision (%) vs. efficiency (GFLOPs) on ActivityNet.** With a ResNet classifier (*left*), OCSampler [24] is the second best after ours. With TimeSformer in (*right*), however, it even underperforms than the uniform sampling. On the other hand, our approach outperforms all baselines with both ResNet50 and TimeSformer.

time, namely throughput, by measuring the video processing speed per second. The experiments are conducted using ResNet50 on the ActivityNet-v1.3 dataset, and all experiments are performed on a single NVIDIA Xp GPU. As seen in Tab. II, we demonstrate improved accuracy of our proposed method (SOSampler) over existing methods, achieving a reduction of approximately 16.3% in GFLOPs and a 15% enhancement in throughput.

Methods	mAP	GFLOPs	Throughput (Videos/s)
AdaFrame [18]	71.5%	79.0	6.4
FrameExit [18]	76.1%	26.1	19.1
AR-Net [26]	73.8%	33.4	23.1
AdaFocus [18]	75.0%	26.6	44.9
OCSampler [24]	77.2%	25.8	107.7
SOSampler (ours)	77.3%	21.6	123.9

Table II: **Comparison of computational overhead** in GFLOPs and throughput. (ResNet50 on ActivityNet)

B.3 Performance and Efficiency Curve

In Fig. I(left), we compare our approach to existing methods with varying computational costs, with a varied number of sampled frames $N = 2, 3, 4, 6$ on a ResNet50 [12] classifier. Our method leads all other compared methods, using significantly lower computational cost than most baseline methods, showing marginal improvement over OCSampler [24].

We additionally conduct a performance and efficiency comparison using the TimeSformer [2] backbone. The experiment, like the one performed on ResNet50, measures the changes in computational cost for $N = 2, 3, 4, 6$ and the comparison is exclusively with OCSampler, previously the highest-performing model. As shown in Fig. I(right), within the TimeSformer architecture, our model significantly improves the performance over OCSampler.

C Sampling Cases

In Sec. 4.1, we introduce π_s and demonstrate that it approximates π_o through Tab. 1 and Tab. 2. By showing that SOSampler, which learns π_o instead of π_s , outperforms existing methods across various datasets and architectures, we demonstrate the effectiveness of π_s .

In this section, for a better understanding of our approach, we visually illustrate multiple examples showing that SOSampler successfully approximates π_s as well as some cases where it does not.

C.1 Illustration of Successful Sampling





									
Riding Camel									
π_o	0	0			0	0	0		0
π_s	0	0			0	0	0		0
S_{SO}	0	0			0	0	0		0
									
Washing Dishes									
π_o	0			0	0	0	0	0	
π_s	0			0	0	0	0	0	
S_{SO}	0			0	0	0	0	0	
									
Braiding Hair									
π_o	0	0	0	0			0	0	
π_s	0	0	0	0			0		0
S_{SO}	0	0	0	0			0		0
									
Peeling Potatoes									
π_o		0	0			0	0	0	0
π_s		0	0			0	0	0	0
S_{SO}		0	0			0	0	0	0

Figure II: Sampling policy comparison with π_s , π_o for success cases of SOSampler.

In Fig. II, we present qualitative examples of successful sampling by SOSampler, comparing the results with those of π_o and π_s . For the “Riding Camel” example, although the 4th and 5th frames feature a camel, they are not selected due to the lack of clear information about riding compared to other scenes. In the “Braiding Hair” example, π_o and π_s choose slightly different frames, with SOSampler following the selection pattern of π_s . In the case

of “Peeling Potatoes”, it is observed that all policies effectively sample only the portions where potatoes appear.

These results demonstrate that π_o and π_s possess similar policies. Additionally, they indicate that SOSampler can effectively learn the policy of π_s .

C.2 Failure Cases

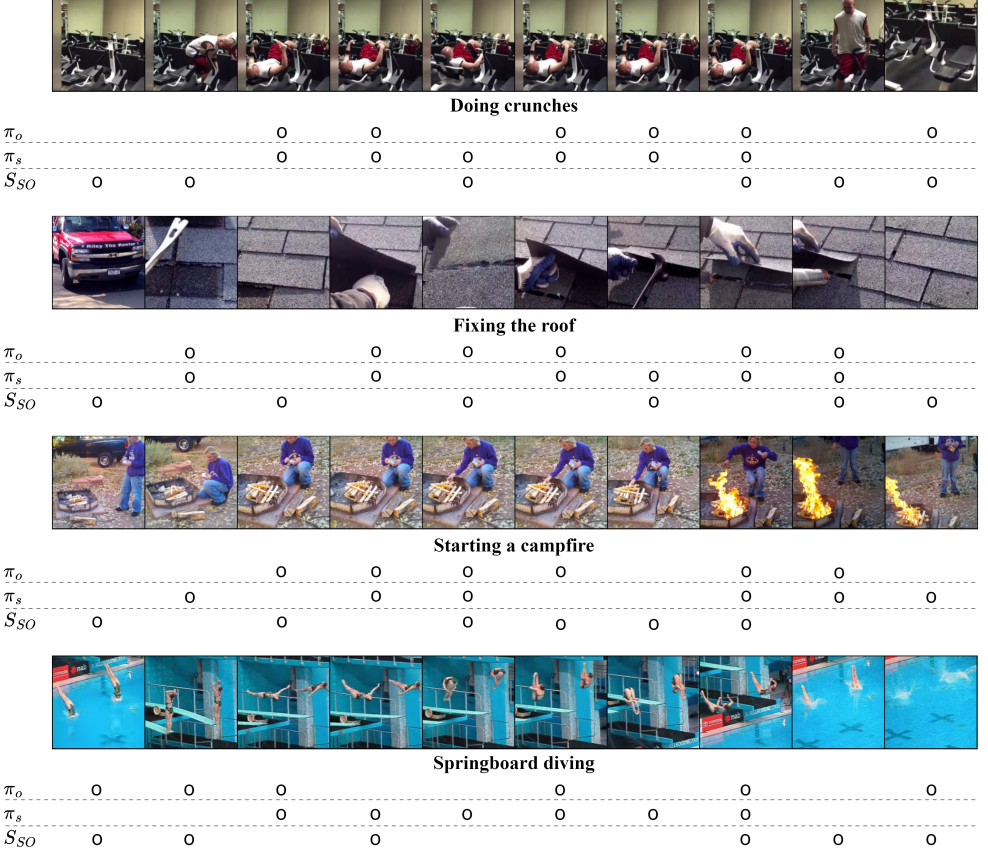


Figure III: Sampling policy comparison with π_s , π_o for failure cases of SOSampler.

As shown in Fig. II, in most cases, S_{SO} demonstrates a sampling policy similar to π_s . In Fig. III, however, we showcase a few scenarios where they significantly differ. In the case of the “Doing Crunches”, π_s effectively samples the segments where a man is performing crunches, while S_{SO} samples scattered frames throughout the video. For the “Fixing the Roof”, π_s appropriately selects scenes of repairing damaged roofs, while S_{SO} chooses unrelated frames as well. In the case of “Starting a Campfire”, S_{SO} seems to summarize the video well, but the sampling policy of π_o indicates that the classifier f_c prefers the scenes of installing firewood and starting the fire. Interestingly, in the “Springboard Diving” example, S_{SO} even appears to better emulate π_o than π_s does.