This WACV 2023 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Stop or Forward: Dynamic Layer Skipping for Efficient Action Recognition

Jonghyeon Seon[1] Jaedong Hwang[2] Jonghwan Mun[3] Bohyung Han[1]
[1]Seoul National University [2]Massachusetts Institute of Technology [3]Kakao Brain
[1]{sunutf, bhhan}@snu.ac.kr [2]jdhwang@mit.edu [3]jason.mun@kakaobrain.com

Abstract

One of the challenges for analyzing video contents (e.g., actions) is high computational cost, especially for the tasks that require processing densely sampled frames in a long video. We present a novel efficient action recognition algorithm, which allocates computational resources adaptively to individual frames depending on their relevance and significance. Specifically, our algorithm adopts LSTM-based policy modules and sequentially estimates the usefulness of each frame based on their intermediate representations. If a certain frame is unlikely to be helpful for recognizing actions, our model stops forwarding the features to the rest of the layers and starts to consider the next sampled frame. We further reduce the computational cost of our approach by introducing a simple yet effective early termination strategy during the inference procedure. We evaluate the proposed algorithm on three public benchmarks: ActivityNet-v1.3, Mini-Kinetics, and THUMOS'14. Our experiments show that the proposed approach achieves outstanding trade-off between accuracy and efficiency in action recognition.

1. Introduction

As the amount of videos stored in private and public repositories explodes, there has been a growing interest in analyzing and understanding video content in recent years. Action recognition is one of the most primitive tasks in video understanding, and existing approaches [2, 8, 19, 20, 31, 35] often perform dense prediction over a sequence of frames or clips, *i.e.*, short time intervals of a few seconds. To be specific, they extract features from a set of frames (or clips) in a video via a sliding window scheme, process individual frames using a deep neural network, and finally identify an action label by aggregating the prediction scores of all the frames. Such a costly procedure is impractical in real-world scenarios, where the algorithms need to run in resource-limited environments. Reducing computational costs for video analysis is critical in practical applications.

The key idea of efficient action recognition comes from the intuition that all frames in a video are not equally important. Consequently, models do not need to observe all frames and can skip *irrelevant* or *repetitive* frames without any penalty. Several action recognition techniques pursue efficient processing through frame selection (or sampling) [6, 12, 16, 43] or adaptive resource allocation [21]. The frame selection methods utilize an external network to 1) determine whether the current frame is worth forwarding to full backbone models for inference [12, 16, 19] or 2) sample the position of the next input frame while skipping redundant ones [6, 43]. On the other hand, adaptive computation models process frames in multiple resolutions using networks with different capacities [21] or select cropped patches in frames [37], depending on their estimated importance. However, many frame selection methods are only available on recorded videos, due to the use of global memory [43] or preprocessed video features [6, 16]. These properties hinder the applicability in online processing environments such as real-time surveillance systems and streaming services.

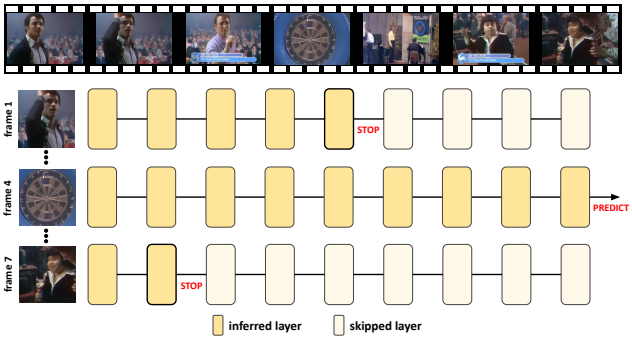

Figure 1: The overview of our approach with an example video in the *Throwing Darts* class of the ActivityNet-v1.3 dataset. Our model achieves efficient action recognition by skipping less important frames in the middle of the classification network adaptively, *e.g.*, frame 1 and 7 in this figure. The proposed algorithm reduces computational costs as many layers as it skips.

We propose an adaptive computation algorithm based on dynamic layer skipping for efficient action recognition, referred to as SoF-Net (Stop-or-Forward Network), which re-

duces the computational cost by skipping layers at inference time depending on their importance. Our approach is available on online processing as illustrated in Figure 1. The decision-making for layer skipping relies on the policy modules implemented with a set of LSTMs, which are applied to several intermediate layers in the backbone network. The policy modules enable the classification network to make the final predictions based only on a fraction of input frames while skipping irrelevant or repetitive frames in the middle of forwarding processes. We also introduce a simple but effective way to further reduce the computational cost by terminating inference completely before observing all frames based on prediction scores and their confidence. To the best of our knowledge, this is the first attempt to show how to determine the frame's usefulness in the intermediate layer of a network.

The contribution of this paper is summarized as follows:

- We propose an adaptive resource allocation method for efficient action recognition, which reduces the computational cost by skipping less important frames in the middle of forwarding processes and terminating the inference procedure even before observing all frames.
- We introduce a simple but effective self-supervised learning method via learning LSTM-based policy modules that are responsible for the proposed dynamic layer skipping.
- Our approach achieves an excellent trade-off between accuracy and efficiency on multiple benchmarks including ActivityNet-v1.3, Mini-Kinetics, and THUMOS'14.

2. Related Works

This section overviews efficient action recognition methods and adaptive computation techniques for deep neural networks.

2.1. Efficient action recognition methods

Efficient networks Although 3D CNN architectures [2, 31] have widely been used for video understanding, they suffer from large computational costs incurred by the complex operations to handle spatio-temporal information jointly. To tackle this challenge, some approaches rely on the models based on 2D CNNs [35] or their extensions by incorporating temporal shift modules [20] or using temporal difference [46]. Another line of research for designing lightweight action recognition models is to decompose spatio-temporal information into multiple subspaces, *e.g.*, spatial and temporal information [4, 11, 17, 18, 23, 25, 32, 33, 44]. Although efficient networks are successful in action recognition, they are limited to focusing on architecture designs without considering the characteristics of input videos.

Efficient frame selection Some action recognition techniques achieve efficiency by adaptively selecting a subset of frames in an input video for prediction [1, 15]. These approaches employ either a lightweight network [16] or multiple reinforcement learning agents [40] to identify the frames for passing into the full backbone models. AdaFrame [43], for example, chooses the next frame for observation using the global context obtained from a designated neural network [28]. FrameExit [7] presents an early termination method via non-sequential processing of frames. The main drawback of these methods is that they are designed for offline processing by default. On the other hand, there exist several methods that perform sequential decision-making during inference [3, 6, 21, 42, 43, 45]. LiteEval [42] employs coarse and fine LSTMs to propagate features, where a conditional gating module determines when to allocate more resources for feature computation. AR-Net [21] utilizes a lightweight policy network that selects proper resolutions of input frames and corresponding classification networks to reduce computation for unimportant frames. AdaFocus [37] also reduces costs by applying a lightweight network, which selects cropped patches from each frame. OCSampler [19] employ reinforcement learning to select a fixed number of frames from candidates to reduce computation. Unlike previous approaches, our algorithm exploits intermediate representations in a CNN to stop processing unnecessary frames and reduce computational costs.

2.2. Adaptive computation techniques

Adaptive computation for reducing computational cost and improving performance is used in many areas such as image recognition [26, 38, 41], natural language processing [29] and semantic correspondence [22]. Most approaches [22, 34, 41] choose active layers based on the importance of input frames. Some methods adopt a policy network that decides whether to drop or keep each layer block for the applications of image classification by using reinforcement learning [36, 41], or Gumbel-Softmax [34]. ACT [5] adaptively selects a subset of layers in each residual block of ResNet [9] for processing based on so-called halting scores. A Similar approach has recently been proposed in a transformer-based model, DynamicViT [26], which chooses salient tokens by inserting a prediction module into the transformer.

In natural language processing, Skim-RNN [29] makes a decision for sending a current input word to a small RNN for skimming or big one for proofreading. On the other hand, [10, 30, 38] share the idea with anytime prediction techniques and reduce computation costs successfully. They incorporate classifiers into multiple layers to measure

confidence and optionally calculate budgets, and make predictions before processing all the layers in the networks depending on the outputs of the classifiers. However, they are limited to being designed for images and have not been applied to videos yet.

3. Proposed Method

3.1. Overview

Given a video with T frames, $V = \{v_1, v_2, ..., v_T\}$, the objective of efficient action recognition is to identify an action label y for the video with a low computational cost in terms of GFLOPS, memory usage, and so on. For efficient action recognition, we adopt the adaptive resource allocation framework that aims to allocate a different amount of computational cost to each frame depending on its importance. For the adaptive resource allocation, our main assumption is that irrelevant or redundant frames can often be identified using the representations in lower layers. Based on the assumption, we propose a dynamic layer-skipping strategy that allows the model to stop the evaluation of input frames at intermediate layers adaptively. To this end, we employ policy modules to control the inference flow of a backbone network, where the backbone network and policy modules are optimized jointly to minimize costs for unnecessary frames via our layer-wise decision-making technique and maximize prediction accuracy based only on the relevant frames.

3.2. Stop-or-Forward Network (SoF-Net)

The overall framework of the proposed approach is illustrated in Figure 2. SoF-Net consists of a backbone network with L layers and multiple LSTM-based policy modules; the backbone network computes a visual feature at each layer and an associated policy module is in charge of making a decision—stopping inference (*i.e.*, skipping the input frame) or forwarding the feature to the subsequent layers. Note that the policy modules are employed in the backbone network after pre-selected $N(< L)$ layers, which are aligned with the stages of modern CNNs—stacked building blocks with identical structures, *e.g.*, ResBlock in ResNet [9]. Thus, we use N policy modules and denote an index of layer with the n^{th} policy module by l_n.

Given T frames sampled uniformly from an input video, our model processes the frames sequentially. At the l_n^{th} layer of the t^{th} frame, we first obtain a visual feature $x_t^{l_n}$ from the backbone network and perform the average pooling (AvgPool) on it. Then, the LSTM-based policy module generates a probability distribution over two options using the current pooled visual feature and a hidden state that contains historical information of non-skipped frames, and then samples an action using the Gumbel-Softmax trick [13] that makes the sampling operation differentiable. The decision process by the n^{th} policy module is summarized as follows:

$$\bar{x}_t^{(n)} = \text{AvgPool}(x_t^{l_n}), \tag{1}$$

$$h_t^{(n)} = \text{LSTM}^{(n)}(W_x^{(n)} \bar{x}_t^{(n)}, h_{\bar{t}}^{(n)}), \tag{2}$$

$$q_t^{(n)} = W_g^{(n)} h_t^{(n)}, \tag{3}$$

$$g_t^{(n)} = \text{Gumbel-Softmax}(q_t^{(n)}), \tag{4}$$

where $\bar{x}_t^{(n)}$ is the average-pooled visual feature for the n^{th} LSTM at the t^{th} frame while $\bar{t}$ and $h_t^{(n)}$ denote the index of the last non-skipped frame and the hidden state of the n^{th} LSTM for the frame, respectively. The learnable embedding matrices, W_x and W_g, correspond to FC and Gating FC layers in Figure 2, respectively. The sampled action, $g_t^{(n)} \in \{0, 1\}$, from Gumbel-Softmax$(\cdot)$ represents either *stop* or *forward* for the n^{th} policy; if the sampled action is 0, we skip the frame under consideration and continue the inference otherwise.

Among all the T frames, we compute the logit z_t at the t^{th} frame only if the frame is not skipped, which is given by

$$z_t = \text{MLP}(x_t^L), \tag{5}$$

where MLP$(\cdot)$ denotes a multi-layer perceptron. The final prediction p^T is obtained by applying the softmax function to the aggregated logit as follows:

$$p^T = \text{Softmax}\left(\frac{1}{\sum_t s_t} \sum_{t=1}^{T} s_t z_t\right), \tag{6}$$

where $s_t \equiv \prod_n g_t^{(n)}$ is an indicator variable of frame skipping.

3.2.1 Conditional early termination

In addition to the aforementioned dynamic layer skipping technique, we introduce a strategy to terminate the inference procedure without observing all the T frames in the input video. This is motivated by the fact that people can recognize visual content after watching only a few early frames in a video if a sufficient amount of evidence is collected. Based on the inspiration, we terminate the inference early and report the final classification result at the point when the prediction for non-skipped frames becomes higher than a threshold, Formally, the inference is terminated if the following condition meets:

$$\max_c \; p_c^{\overline{T}} > \rho, \tag{7}$$

where $p_c^{\overline{T}}$ denotes the probability of action label c after processing the $\overline{T}^{\text{th}}$ frame as in Eq. (6) and ρ is a threshold of the prediction probability. Note that $\overline{T} > T_{\text{min}}$, where T_{min}

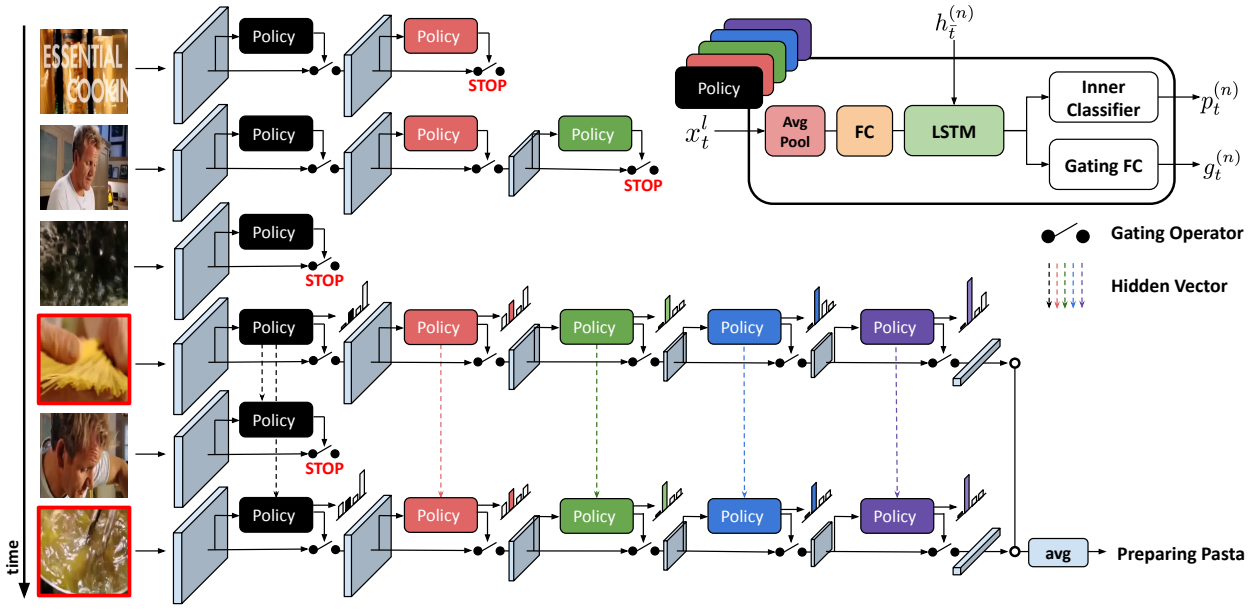

Figure 2: A run-time procedure of the proposed algorithm. At several predefined intermediate layers of the backbone classification model, we employ LSTM-based policy modules to determine whether it stops or continues the inference procedure. The policy modules are learned to predict stop signals for irrelevant or redundant frames while encouraging the classification network to go through all layers for important frames. The final prediction for a video is obtained by aggregating the prediction scores over the fully processed frames.

is the minimum number of frames for early termination to avoid too hasty decision.

The proposed early termination scheme is incorporated on top of the dynamic layer skipping technique, which leads to a desirable combination of architectural and temporal optimization for efficient action recognition. Also, contrary to [7], our full algorithm processes video frames sequentially and runs online.

3.3. Training

We train our model using three loss terms, which include 1) action classification loss $\mathcal{L}_{\text{cls}}$, 2) efficiency loss $\mathcal{L}_{\text{eff}}$, and 3) policy guidance loss $\mathcal{L}_{\text{pg}}$. The total loss is given by

$$\mathcal{L} = \alpha\mathcal{L}_{\text{cls}} + (1-\alpha)\mathcal{L}_{\text{eff}} + \mathcal{L}_{\text{pg}}, \tag{8}$$

where α balances the trade-off between recognition accuracy and computational cost.

3.3.1 Action classification loss

To predict a correct action label y that is represented by an one-hot encoded vector, the backbone network is learned using a standard cross-entropy loss ($\mathcal{L}_{\text{CE}}$) as follows:

$$\mathcal{L}_{\text{cls}} = \mathcal{L}_{\text{CE}}(p, y) = -\sum_{c\in\mathcal{C}} y_c \log p_c, \tag{9}$$

where $\mathcal{C}$ indicates the label set.

3.3.2 Efficiency loss

To make our model run efficiently, the policy modules are learned to minimize the amount of the overall computation (GFLOPS) for an input video with T frames. For this purpose, we construct a lookup table that stores expected GFLOPS for each policy module; the expected GFLOPS of the n^{th} policy module is defined by GFLOPS for inferring the remaining layers after the l_n^{th} layer, which is given by

$$f_{\text{lookup}}(n) = f_{\text{GFLOPS}}(L) - f_{\text{GFLOPS}}(l_n), \tag{10}$$

where $f_{\text{GFLOPS}}(l)$ denotes computational cost in terms of GFLOPS when inferring until the l^{th} layer and $f_{\text{lookup}}(\cdot)$ is a lookup table value. The efficiency loss based on the expected GFLOPS over T frames and N policy modules is as follows:

$$\mathcal{L}_{\text{eff}} = \frac{1}{T\cdot N}\sum_{t=1}^{T}\sum_{n=1}^{N} f_{\text{lookup}}(n) g_t^{(n)}. \tag{11}$$

3.3.3 Policy guidance loss

To learn better policy modules, we incorporate an additional inner classifier as guidance for each module. The policy guidance loss is defined as

$$\mathcal{L}_{\text{pg}} = \beta\mathcal{L}_{\text{cls}}^{\text{inner}} + (1-\beta)\mathcal{L}_{\text{self}}, \tag{12}$$

where $\mathcal{L}_{\text{cls}}^{\text{inner}}$ and $\mathcal{L}_{\text{self}}$ denote inner-cassification loss and self-supervision loss, respectively, and β controls the trade-off between recognition accuracy and computational cost.

Inner classification loss To generate self-supervision for policy modules, we train additional inner classifiers attached to individual policy modules as illustrated in Figure 2. The inner classifiers are learned by the cross-entropy loss only when the associated policy module generates a forward signal, *i.e.*, $g_t^{(n)} = 1$, which is given by

$$\mathcal{L}_{\text{cls}}^{\text{inner}} = \frac{1}{\sum_t \sum_n g_t^{(n)}} \sum_{t=1}^{T} \sum_{n=1}^{N} g_t^{(n)} \mathcal{L}_{\text{CE}}(p_t^{(n)}, y), \quad (13)$$

where $p_t^{(n)} = W_{\text{cls}}^{(n)} h_t^{(n)}$ is a predictive distribution based on a learnable embedding matrix $W_{\text{cls}}^{(n)}$ in the n^{th} policy module at the t^{th} frame.

Self-supervision loss We train the policy module using the pseudo-label $\hat{g}_t^{(n)}$, which is estimated by the inner classifier. Specifically, if the classification score from the inner classifier increases progressively over the layers, we continue to observe the subsequent layers for which the pseudo-label of the corresponding policy module is defined by

$$\hat{g}_t^{(n)} = \begin{cases} 1 & \text{if } p_t^{(n+1)}(y) > p_t^{(n)}(y) \\ 0 & \text{otherwise} \end{cases}. \quad (14)$$

Then the self-supervision loss to train the policy modules except the last one is given by

$$\mathcal{L}_{\text{self}} = \frac{1}{T \cdot (N-1)} \sum_{t=1}^{T} \sum_{i=1}^{N-1} \mathcal{L}_{\text{BCE}}(g_t^{(n)}, \hat{g}_t^{(n)}), \quad (15)$$

where $\mathcal{L}_{\text{BCE}}$ denotes a binary cross-entropy loss function with two vectorized input values.

4. Experiments

We evaluate the proposed approachon three standard benchmarks and report the results.

4.1. Experimental setup

Datasets We conduct experiments on three action recognition datasets: ActivityNet-v1.3 [1], Mini-Kinetics [2], and THUMOS'14 [14]. ActivityNet-v1.3 is composed of untrimmed long videos, which are divided into 10,024 training and 4,926 validation examples for 200 action classes. The average duration of the videos is 117 seconds. Mini-Kinetics contains 200 classes and 131,082 trimmed videos, 121,215 for training and 9,867 for testing, sampled from the original Kinetics dataset [2]. The average length of the videos is 10 seconds. We train models using the training set while evaluating algorithms on the validation or test splits on ActivityNet-v1.3 and Mini-Kinetics. THUMOS'14 contains videos over 24 hours in 101 different sport activities. The validation and the test sets contain 1,010 and 1,574 untrimmed videos, respectively, and the validation set is used for training.

Evaluation metrics For evaluation, we adopt the mean Average Precision (mAP) for ActivityNet-v1.3 and THUMOS'14 while using the top-1 accuracy for Mini-Kinetics. On the other hand, to compare efficiency of models, we measure GFLOPS per frame (GFLOPS/f), GFLOPS per video (GFLOPS/V), and runtime per video (Runtime/V).

4.2. Implementation details

As a backbone network, we adopt ResNet-50 [9] pre-trained on the ImageNet [27] dataset. We uniformly sample $T = 16$ frames from each video during both training and testing and resize to 168×168 resolutions. The policy module is attached to the end of each residual block (*i.e.*, res1,res2, res3, res4, and res5), thus we use $N = 5$ policy modules. The policy modules are defined by a single-layer LSTM with a 512-dimensional hidden state. We set the initial temperature of Gumbel-Softmax to 5, and gradually anneal it with an exponential decay factor of -0.045 in every epoch following [13]. We set the coefficients of loss terms as $\alpha = 0.9$ and $\beta = 0.9$ in our training. For conditional early termination, the thresholds, ρ and T_{min}, are set to 0.999 and 3, respectively.

We train the backbone and policy modules using the SGD optimizer with an initial learning rate of 0.001, weight decay of 0.00001, and momentum of 0.9. Note that the learning rate is reduced to 0.0001 after 30 epochs. Since a random policy at initial training steps would hinder the learning backbone network, we train our algorithm in two stages. We first warm up the backbone network for 15 epochs while fixing the policy modules, and start to train the entire network including the policy modules for additional 60 epochs. Our model is implemented with PyTorch [24] and all models are trained in 4 Titan XP GPUs with a mini-batch size of 6 videos per GPU. Note that the results of TSN [35] are reproduced with the same hyper-parameters as SoF-Net.

To measure the runtimes of TSN [35], AR-Net [21], AdaFocus [37] and SoF-Net, we test each model in the same environment setup, with 16 uniformly sampled frames from 4,921 videos in the ActivityNet-V1.3 validation set and batch size of 1, and a single GPU (NVIDIA Titan XP) and CPU (Intel® Xeon® CPU E5-2620 v4 @ 2.10GHz). We report the average runtime from five runs.

Table 1: Performance comparison with the state-of-the-art methods on ActivityNet-v1.3 and Mini-Kinetics. Note that first five methods run offline while the others including SoF-Net are online algorithms. Our implementation of FrameExit (online) does not use its original frame sampling strategy, but sequentially takes input frames given by uniform sampling. MV2 and R# denote MobileNet-V2 [28] and ResNet with the number of layers, respectively, and T is the default number of input frames to run each algorithm . Results of other methods are copied from [19] while † denotes our reproduction. The best results are in bold.

Type	Method	Backbone	Resolution	T	ActivityNet-v1.3			Mini-Kinetics		
					mAP	GFLOPS/f	GFLOPS/V	Top1	GFLOPS/f	GFLOPS/V
Offline	AdaFrame [43]	MV2+R101	224	25	71.5	3.16	79.0	-	-	-
	ListenToLook [6]	MV2+R152	224	16	72.3	5.09	81.4	-	-	-
	SCSampler [16]	MV2+R50	224	16	72.9	2.62	42.0	70.8	2.62	42.0
	FrameExit [7]	R50	224	10	76.1	2.61	26.1	72.8	**1.97**	**19.7**
	FrameExit [7]	R50	224	16	76.1†	**2.19**†	35.1†	-	-	-
	OCSampler [19]	MV2+R50	224	10	**77.2**	2.58	**25.8**	**73.7**	2.16	21.6
Online	LiteEval [42]	MV2+R101	224	25	72.7	3.80	95.1	61.0	3.96	99.0
	AR-Net [21]	MV2+R50/R32/R18	224/168/112	16	73.8	2.09	33.5	71.7	2.00	32.0
	FrameExit (online) [7]	R50	224	10	73.7†	2.76†	27.6†	-	-	-
	AdaFocus [37]	MV2+R50	128	16	75.0	**1.66**	**26.6**	72.2	**1.64**	**26.3**
	SoF-Net (ours)	R50	168	16	**75.3**	1.71	27.4	**72.8**	1.75	28.0

Table 2: Action recognition results on THUMOS'14.

Method	mAP	GFLOPS/f	GFLOPS/V
TSN [35]	46.6	4.12	65.9
AR-Net [21]	47.4	1.67	26.7
SoF-Net	**47.8**	**1.60**	**25.6**

Table 3: Comparison of runtime and frame usage ratio on the ActivityNet-v1.3 validation set. The numbers in SoF-Net denote input image sizes.

Method	Runtime/V (ms)	Total Runtime (s)	Frame Usage (%)
TSN [35]	110.5	543.6	100.0
AR-Net [21]	120.8	594.6	70.1
AdaFocus [37]	165.1	812.3	100.0
SoF-Net (168)	**74.6**	**367.2**	**54.3**
SoF-Net (192)	82.0	403.4	61.0
SoF-Net (224)	83.9	412.8	59.0

4.3. Comparison with other methods

We compare the proposed SoF-Net with the state-of-the-art efficient action recognition techniques in two branches: frame selection methods such as AdaFrame [43], LiteEval [42], ListenToLook [6] SCSampler [16], FrameExit [7], and OCSampler [19], and adaptive computation framework such as AR-Net [21] and AdaFocus [37]. Table 1 and 2 summarize the results on ActivityNet-v1.3, Mini-Kinetics, and THUMOS'14. In THUMOS'14, we use 224×224 images and sample 16 frames from each video for training a TSN [35] model. We also compare our method with an online version of FrameExit [7] by removing its heuristic frame sampling strategy—observing frames from the center to the sides temporally. Our method outperforms all competing methods in accuracy with smaller or comparable computational costs and parameters.

To compare the efficiency of algorithms, we additionally present the runtime and frame usage ratio in Table 3 of our approaches and other methods. The results show that SoF-Net is faster in inference than its counterparts by using fewer frames regardless of input resolution; we only use less than 60% of frames to predict action in videos. Note that AR-Net [21] not only uses 70% of frames but also uses four backbone networks and four resolutions for each backbone network to process frames, resulting in high latency. AdaFocus [37] appears to be efficient in terms of GFLOPS when using the 128x128 cropped images as its inputs but turns out to have the longest runtime. These runtime comparisons indicate that our model is more appropriate than other methods for being applied to practical problems involving online processing requirements.

We believe that the outstanding performance of SoF-Net is mainly due to its unique structure. SoF-Net has a simple procedure based on a single backbone network and employs the features in multiple semantic levels to distinguish redundant or noisy frames for the frame selection. This attribute makes the proposed approach more powerful than other methods that rely on a single pre-defined semantic level for the decision.

4.4. Discussion

For a better understanding of our algorithm, we perform an in-depth analysis on the ActivityNet-v1.3 dataset.

Table 4: Ablation study for individual components in our algorithm, tested on the ActivityNet-v1.3 validation set.

Policy module	Temporal modeling	Policy guidance	Early term.	mAP (%)	GFLOPS/V
-	-	-	-	73.7	40.3
✓	-	-	-	74.3	32.2
✓	✓	-	-	75.0	31.6
✓	-	✓	-	74.7	32.6
✓	✓	✓	-	**75.6**	31.7
✓	✓	✓	✓	75.3	**27.4**

Table 5: Performance comparisons by varying input sizes on the ActivityNet-v1.3 validation set.

Resolution	w/o Early termination		w/ Early termination	
	mAP	GFLOPS (f/V)	mAP	GFLOPS (f/V)
168×168	75.6	1.98 / 31.7	75.3	1.71 / 27.4
192×192	76.4	2.73 / 43.7	76.3	2.34 / 37.4
224×224	77.1	3.48 / 55.7	76.9	3.04 / 48.6

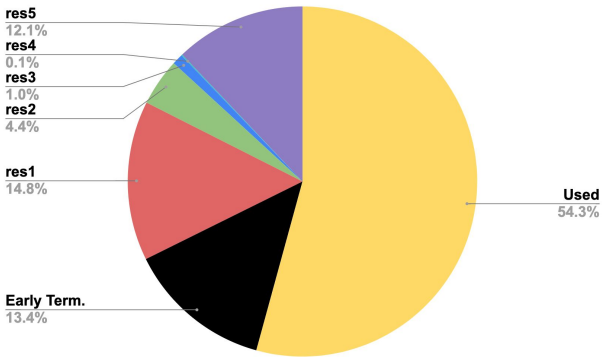

Figure 3: Statistics of frame usages. We show the ratio of the frames that stop forwarding by their stop-decision locations as a form of 'res#.' 'Early Term.' denotes the ratio of the frames skipped by early termination, and 'Used' means the frame ratio for full inference. The numbers in gray color indicates the percentage of each category.

Analysis of our model We perform ablation studies to investigate the contributions of individual components in our algorithm. In this experiment, we train the four variants of our models, where we add individual modules in the sequence of temporal modeling, policy module, policy guidance with self-supervision, and early termination. The policy module without temporal modeling is implemented by replacing LSTM with an FC layer. Table 4 summarizes the results, where we observe the followings. First, the result without temporal modeling implies that the historical information of non-skipped frames is crucial to improve both accuracy and efficiency. Second, the application of the policy module provides 21.3% (40.3 GFLOPS/V to 31.7 GFLOPS/V) efficiency improvement by skipping redundant frames using early layer skipping. Third, the self-supervision obtained from inner classifiers helps the policy modules identify noisy frames, leading to accuracy improvement. Finally, the early termination strategy indeed makes SoF-Net more efficient.

Frame skipping ratio Figure 3 illustrates the statistics of the decisions made by policy modules. Overall, our policy modules learn to use 54.3% of frames while skipping 32.3% on average, where the first and last policy modules, corresponding to res1 and res5, are two common locations, where skipping decisions are made. Frame skipping at the last ResBlock enhances accuracy by preventing confusing frames from being involved in inference. Besides frame skipping, early termination of frames improves efficiency greatly by skipping 13.4% of frames.

Input resolutions We train SoF-Net on various sizes of input frames (168, 192, 224) with and without early termination. As shown in Table 5, the accuracy of the model improves as the size of an input frame increases while the corresponding computation cost also increases. With a resolution of 224×224, our model achieves higher accuracy than the state-of-the-art offline algorithm. In all resolutions, early termination consistently reduces computations with negligible accuracy drops.

4.5. Qualitative Analysis

For a better understanding of how SoF-Net works, we present input frames and their decision-making results in Figure 4. For each example, the bottom row illustrates whether each frame is used for prediction or skipped; res# means that the model stops prediction at the #-th policy module and decides to skip the frame, and the exit indicates frame skipping by 'early termination' while the frames considered to be important by the policy modules are represented as the original frames. The blue bars at the bottom of each case denote action localization annotations provided in the ActivityNet-v1.3 dataset. Note that, considering action localization annotation, SoF-Net effectively captures important frames and skips repetitive or irrelevant frames.

5. Conclusion

We presented a novel efficient action recognition algorithm, SoF-Net, which allocates adaptive computational resources for individual frames based on their importance. Specifically, the policy module in each layer decides to stop forwarding the current frame to the following layers and filter out less important frames, reducing computational costs and improving recognition performance. The module is trained by the efficiency loss and the policy guidance loss

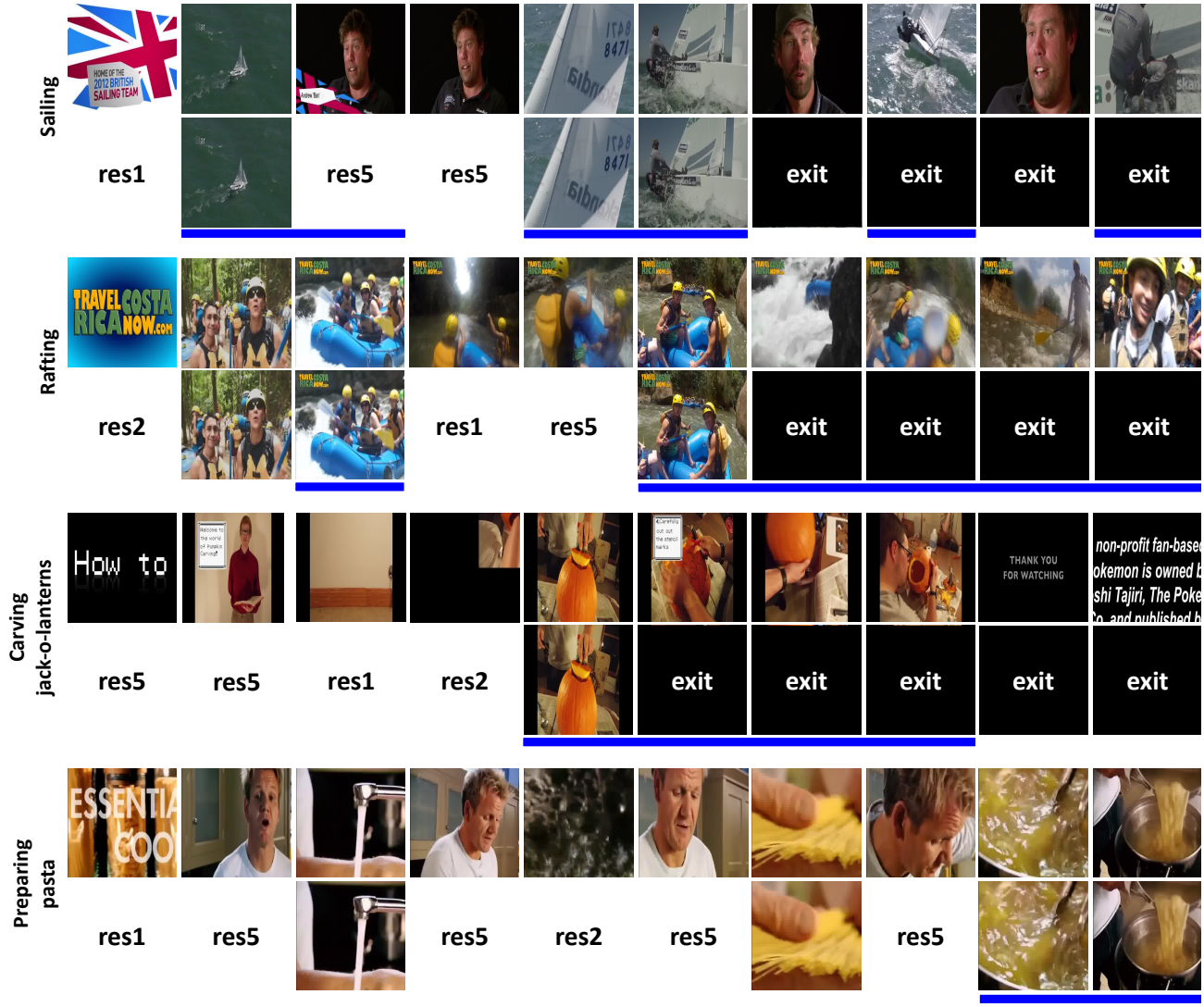

Figure 4: Visualization of the decision-making results by SoF-Net. For each example, the top row shows the original input frames and the bottom row illustrates how the frames are processed in SoF-Net. The skipped frames specifies the position of stopped layer (res#) while we present the original frames for the ones used for prediction. The black box with a word 'exit' indicates the frames that are not observed at all due to early termination, and the blue bar denotes the temporal action localization ground-truths indicating the relevance of the frame to the target action.

by comparison of classification scores of inner classifiers in the current and next layers. Moreover, we also employ a simple yet effective early termination strategy that decides to terminate the inference of a given video.

In summary, SoF-Net has a simple online (sequential) procedure for efficient prediction without using multiple backbone networks (*e.g.*, AR-Net [21], LiteEval [42], OC-Sampler [19]) or adopting offline prediction relying on global memory (*e.g.*, AdaFrame [43]). SoF-Net employs representations in the multiple intermediate layers for frame selection, effectively identifying potential redundancy or noise in various semantic levels. This property makes the proposed approach more potent than other methods relying on a single pre-defined semantic level for the decision. To the best of our knowledge, this is the first attempt to show how to determine the frame's usefulness in the intermediate layer of a network.

Acknowledgements This research was supported in part by Samsung Advanced Institute of Technology and the National Research Foundation (NRF) funded by the Korean government (MSIT) [No. 2021M3A9E4080782, No. 2022R1A5A708390811].

References

[1] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[3] Hehe Fan, Zhongwen Xu, Linchao Zhu, Chenggang Yan, Jianjun Ge, and Yi Yang. Watching a small portion could be as good as watching all: Towards efficient video classification. In *IJCAI*, 2018.

[4] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020.

[5] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *CVPR*, 2017.

[6] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. In *CVPR*, 2020.

[7] Amir Ghodrati, Babak Ehteshami Bejnordi, and Amirhossein Habibian. FrameExit: Conditional Early Exiting for Efficient Video Recognition. In *CVPR*, 2021.

[8] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[10] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018.

[11] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *CVPR*, 2019.

[12] Noureldien Hussein, Mihir Jain, and Babak Ehteshami Bejnordi. TimeGate: Conditional Gating of Segments in Long-range Activities. *arXiv preprint arXiv:2004.01808*, 2020.

[13] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[14] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

[15] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *TPAMI*, 40(2):352–364, 2017.

[16] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

[17] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Collaborative spatiotemporal feature learning for video action recognition. In *CVPR*, 2019.

[18] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020.

[19] Jintao Lin, Haodong Duan, Kai Chen, Dahua Lin, and Limin Wang. Ocsampler: Compressing videos to one clip with single-step sampling. In *CVPR*, 2022.

[20] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

[21] 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.

[22] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Learning to compose hypercolumns for visual correspondence. In *ECCV*, 2020.

[23] Bowen Pan, Rameswar Panda, Camilo Fosco, Chung-Ching Lin, Alex Andonian, Yue Meng, Kate Saenko, Aude Oliva, and Rogerio Feris. Va-red ^{2}: Video adaptive redundancy reduction. *ICLR*, 2021.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[25] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017.

[26] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Neurips*, 2021.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[29] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. In *ICLR*, 2018.

[30] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early existing from deep neural networks. In *ICPR*, 2016.

[31] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[32] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

[33] 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.

[34] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018.

[35] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment

networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[36] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018.

[37] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. In *ICCV*, 2021.

[38] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *Neruips*, 2020.

[39] 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.

[40] 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.

[41] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018.

[42] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *NeurIPS*, 2019.

[43] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *CVPR*, 2019.

[44] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018.

[45] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[46] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.