TAPS: Temporal Attention-based Pruning and Scaling for Efficient Video Action Recognition

Yonatan Dinai[1⋆][0009−0007−8994−054X], Avraham Raviv[1⋆][0000−0002−4428−0505], Nimrod Harel[1][0009−0000−1706−2695], Donghoon Kim[2][0009−0008−3334−309X], Ishay Goldin[1][0009−0003−7883−032X], and Niv Zehngut[1][0009−0000−5623−8040]

[1] Samsung Israel Research Center (SIRC)
[2] Samsung Semiconductor
{yonatan.dina, avraham.r, nimrod.harel, dhoon.kim, ishay.goldin, niv.z}@samsung.com

Abstract. Video neural networks are computationally expensive. For real-time applications they require significant compute resources that are lacking on edge devices. Various methods were proposed to reduce the computational load of neural networks. Among them, dynamic approaches adapt the network architecture, its weights or the input resolution to the content of the input. Our proposed approach, showcased on the task of video action recognition, allows to dynamically reduce computations for a wide range of video processing networks by utilizing the redundancy between frames and channels. A per-layer lightweight policy network is used to make a per-filter decision regarding the filter's importance. Important filters are retained while others are scaled down or entirely skipped. Our method is the first to allow the policy network to gain a broader temporal context considering features aggregated over time. Temporal aggregation is done using self-attention between present, past and future (if available) input tensor descriptors. As demonstrated on a large variety of leading benchmarks such as Something-Something-V2, Mini-Kinetics, Jester and ActivityNet1.3, and over multiple network architectures, our method is able to enhance accuracy or save up to 70% of the FLOPs with no accuracy degradation, outperforming existing dynamic pruning methods by a large margin and setting a new bar for the accuracy-efficiency trade-off allowed by dynamic methods. We release the code and trained models at https://github.com/tapsdyn/TAPS.

Keywords: Action Recognition · Dynamic Pruning · Efficient Inference

1 Introduction

With the exponential growth of video-based applications across a multitude of devices, from smartphones and AR/VR sets to autonomous vehicles and drones, there is an urgent need to optimize the computational complexity, memory footprint, and bandwidth requirements of video processing neural networks. Dedicated

⋆ Equal contribution.

video architectures such as 3D ConvNets [31] and Video Vision Transformers (VVTs) [1] have made great strides in terms of accuracy. Yet, these architectures typically require great computational resources.

Efficient algorithms for video processing aim to strike a balance between computational complexity and accuracy. One prominent approach is adoption of architectures and model reduction techniques from single image problems to the video processing domain; strategies like weight pruning, quantization, Neural Architecture Search (NAS) and efficient design choices [26].

A second approach uses lightweight variants of 3D ConvNets and VVTs. Methods such as [7, 18] retain the ability to capture temporal dynamics while minimizing computational complexity, making them suitable for efficient video analysis on a wide range of devices.

Another effective group of optimization methods is dynamic neural networks. In these methods the network adapts its architecture to each input instance [14]. Dynamic pruning methods, in particular, strive to identify and omit redundancies during inference. These methods vary in terms of the specific types of redundancies they target and the underlying mechanisms they employ. In the context of image-level analysis, redundancies can be observed along the spatial dimension [33] or the channel/layer/block dimension [9]. In video tasks, additional temporal redundancies emerge, such as similar content appearing in multiple frames.

Several recent methods were proposed for dynamic pruning in videos [24, 25]. These methods share a fairly simple dynamic pruning mechanism; a small and efficient policy network that tries, during inference, to find redundant computations that can be omitted. While improving efficiency, previous works that incorporated temporal information as input to the policy network focused on temporally local changes, considering only pairs of consecutive frames. Yet, frame features that are not redundant with respect to the previous frame may still be neglected considering earlier frames. In our work, we overcome this limitation by aggregating a broader range of temporal information into the policy network, enabling a more holistic and informed decision-making process. The lightweight policy network uses a self-attention mechanism to compare and merge information from all frames.

We focus on CNNs as they are more commonly used in real-time video applications. Furthermore, many video transformers rely on CNN backbones to extract low-level spatiotemporal information and then perform higher level reasoning with attention mechanisms [5, 12]. These architectures will benefit from TAPS as well.

In addition, our approach expands the functionality of the policy network. In previous works, the output of the policy network is binary (or ternary [24]), determining which channels are to be computed. In contrast, our policy network predicts a multiplicative scaling factor for each computed channel, retaining or repressing it, increasing network accuracy. The scaling operation is used in conjunction with channel pruning, providing improved accuracy-efficiency trade-offs.

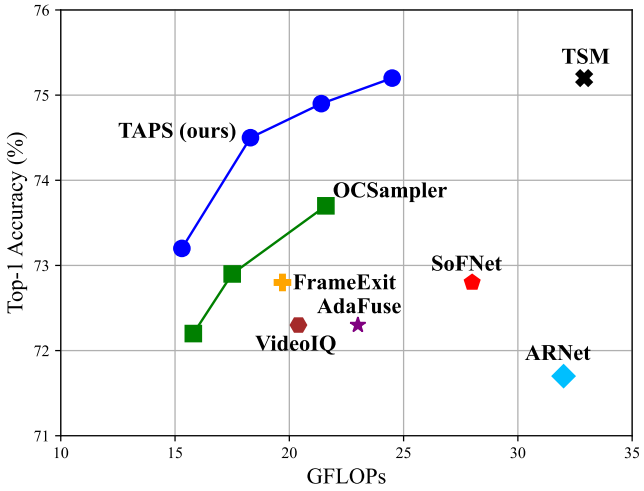

Fig. 1: TAPS compared to state-of-the-art dynamic inference methods for action recognition with ResNet50 on Mini-Kinetics.

Our approach is also easier to implement compared to other dynamic methods. While methods like [24] propose copying channels from past frames, incurring bandwidth and memory footprint increase, we only rely on dynamic skipping of entire filters. We find additional motivation in recent studies which demonstrate the potential of dedicated hardware implementations of dynamic channel and spatial pruning [10].

Extensive experiments conducted on Mini-Kinetics, Jester, Something-Something-V2 and ActivityNet1.3 datasets using multiple network architectures including ResNet [15] and MobileNetV2 [26] showcase the effectiveness of our approach. As shown in Figure 1, on Mini-Kinetics with ResNet50, our method provides multiple accuracy-FLOPs trade-offs that significantly outperform leading dynamic methods, allowing minimal accuracy degradation with respect to the static baseline model, TSM [20]. In summary, the main contributions of our work are as follows:

- We present a novel temporal aggregation-based policy network for dynamic channel pruning of video networks and show that it allows more efficient channel pruning while preserving accuracy.
- We show that temporal aggregation-based policy networks can be used to scale channels, leading to significant improvement in accuracy. By combining channel scaling with pruning computations can be dramatically reduced.
- Our method is the first to use temporal attention in the policy network.

– We set a new state-of-the-art accuracy-efficiency trade-off for dynamic methods on the task of video action recognition. Our method allows significant accuracy improvement with reduced computations, exhibited on four different action recognition benchmarks.

2 Related Work

Video Action Recognition. Standard 2D CNNs have proven effective in various video-related tasks, including action recognition. An early design [27] utilized a two-stream CNN that analyzed both appearance and motion inputs; however, 2D CNNs may be incapable of inferring complex temporal relationships. To overcome this limitation, the use of 3D convolutions was introduced [31], extending the convolution operation to extract features from both the spatial and the temporal dimensions. Subsequent works demonstrated the effectiveness of 3D convolutions over 2D alternatives [3]. More recently, vision transformers have shown even higher accuracy, albeit at a significant computational cost [1].

Efficient Networks. Given the high resource consumption of leading video recognition models, such as 3D convolutions, several studies have explored alternative approaches for temporal modeling that utilize the more efficient 2D convolutions. For instance, Temporal Segment Networks (TSN) [34] derive averaged features from stride-sampled frames in order to model long-range temporal structure. Complementing 2D Convolutions with RNN components, such as LSTM [16], had also been explored [30, 32]. Temporal Shift Module (TSM) [20] equips 2D backbones with a channel shift operation along the temporal dimension, thereby allowing information to be exchanged among neighboring frames.

In parallel, other methods focus on neural network model compression, aiming to reduce resource requirements while minimizing accuracy degradation. These techniques often target the architecture and the number of parameters. Successful model compression techniques include weight and channel pruning, quantization, Neural Architecture Search (NAS), knowledge distillation, transportation-oriented compression [28, 29] and depth reduction [6]. For example, NAS has been used to generate highly efficient 3D video architectures [7].

Dynamic Inference. While the aforementioned methods are static, i.e., the model and inference are independent of the input, dynamic methods such as ours adapt the network architecture to the input. Dynamic methods can be combined with or applied on top of static methods. Typically, they identify redundant components and avoid unnecessary computations. In the context of images, redundancy may occur along the spatial and channel dimensions. For example, Dynamic Channel Pruning [9] turns off channels based on the input, while [33] uses small gating branches to decide which spatial positions should be evaluated.

In videos, temporal redundancy also arises. A straightforward yet effective practice is to skip the processing of less important frames. Methods such as SCSampler [19] and OCSampler [21] propose lightweight models that select important frames for processing using a heavier network. FrameExit [11] proposes

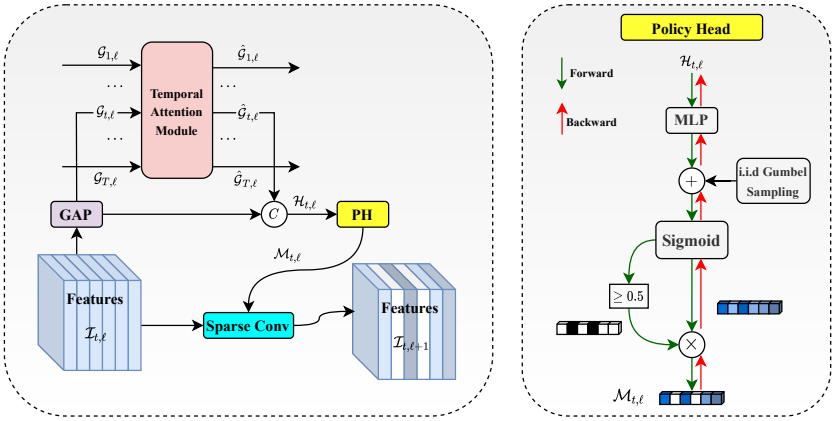

Fig. 2: An illustration of our method for a given frame t and layer ℓ. (a) Global Average Pooling (GAP) descriptor, $\mathcal{G}_{t,\ell}$, is extracted and fed into a temporal attention module (TAM) along with descriptors from other frames. A policy head (PH) then processes $\mathcal{H}_{t,\ell}$, a concatenation of $\mathcal{G}_{t,\ell}$ with the output of the TAM, $\hat{\mathcal{G}}_{t,\ell}$, to produce a mask $\mathcal{M}_{t,\ell}$, leading to a sparse and scaled convolution with a pruned output tensor $\mathcal{I}_{t,\ell+1}$. **(b)** The Policy Head consists of a two-layer MLP followed by Gumbel-SoftMax. The binary mask determines which output channels to skip, preventing nonessential computations, while the rest of the channels are multiplied by the scaling mask.

a conditional early exiting framework to sample fewer frames for simpler videos and more frames for complex ones. LiteEval [37] also processes fewer frames, by using an LSTM-based network that determines which frames will be further processed. AR-Net [23] processes all frames, but at different resolutions, according to their importance. Addressing spatial redundancy, AdaFocus [35] identifies per-frame region-of-interest and ignores the rest of the frame. Focusing on channel redundancy, AdaFuse [24] performs pruning in both the temporal and channel dimensions. By analyzing pairs of consecutive frames, AdaFuse identifies channels that can be skipped altogether or copied from the previous timestamp. Finally, D-STEP [25] combines dynamic inference in the spatial, temporal, and channel domains.

3 Method

Here we propose *Temporal Attention-based Pruning and Scaling, TAPS*. Our dynamic mechanism is founded on small and efficient per-layer policy networks. These networks identify redundancies and predict which filters can be omitted, as well as scale the remaining filters by a multiplicative importance factor. TAPS improves upon the limitations of earlier channel pruning methods for video, first and foremost, by attending features from all past frames (and future frames when processing a video offline). In addition, TAPS generalizes the typical on/off decision with effective soft scaling.

Temporal Attention-based Pruning and Scaling. A general overview of our architecture is shown in Figure 2. Consider a video consisting of T frames and a 2D CNN applied to each frame. For layer ℓ of frame t, denote the input feature map as $\mathcal{I}_{t,\ell} \in \mathbb{R}^{h_\ell \times w_\ell \times c_\ell}$. The output of a standard 2D convolution is $\tilde{\mathcal{I}}_{t,\ell+1} = \varphi(\mathcal{W}_\ell * \mathcal{I}_{t,\ell} + b_\ell)$, where φ denotes the activation function, $\mathcal{W}_\ell \in \mathbb{R}^{k \times k \times c_\ell \times c_{\ell+1}}$ denotes the filters of kernel size $k \times k$ and $b_\ell \in \mathbb{R}^{c_{\ell+1}}$ is the bias. TAPS policy network receives $\mathcal{I}_{t,\ell}$ as input and computes a sparse per-output-channel mask $\mathcal{M}_{t,\ell} \in [0,1]^{c_{\ell+1}}$ such that:

$$\mathcal{I}_{t,\ell+1} = \varphi(\mathcal{M}_{t,\ell} \odot [\mathcal{W}_{t,\ell} * \mathcal{I}_{t,\ell} + b_\ell]) \tag{1}$$

where $\odot$ denotes the channel scaling operator and if $\mathcal{M}_{t,\ell}[i] = 0$, the i^{th} filter is skipped. TAPS policy network first performs Global Average Pooling (GAP) on $\mathcal{I}_{t,\ell}$, resulting in a vector denoted as $\mathcal{G}_{t,\ell}$. This per frame descriptor is the input token to the attention module.

Temporal Attention Module (TAM). This module is designed to capture global dependencies between frames by applying lightweight, multi-head, self-attention operations between all GAP descriptors, $\{\mathcal{G}_{t,\ell}\}_{t\in 1,\ldots,T}$, resulting with temporally aggregated descriptors, $\{\hat{\mathcal{G}}_{t,\ell}\}_{t\in 1,\ldots,T}$, that are used as input to the policy network head. Since a single attention layer accounts for the relations between pairs of frames, we use two consecutive attention layers to capture more complex temporal dependencies and improve performance.

Online Processing Mode. Many works on dynamic inference in videos rely on the availability of all frames prior to processing (offline mode) [19, 21, 38, 39]. Real-time applications, however, often operate online, processing a video stream frame by frame without access to future information. Our attention-based policy networks adapt to both modes. In offline mode, attention spans all GAP descriptors, while in online mode, it considers only current and past descriptors. The attention mechanism can be trained with both past and future information but relies solely on past information during online inference.

Policy Head. The TAM is followed by a policy network head (PH), described in Figure 2(b). We used two fully connected layers separated by batch normalization (BN) and ReLU activation. Next, a Sigmoid function is applied, and during inference, values below 0.5 mark filters that can be omitted. In addition, Sigmoid output is used as a multiplicative factor to dynamically scale the remaining channels according to their importance.

Training Non-Differentiable Policy. The policy network performs a discrete pruning decision which cannot be trained using traditional backpropagation. To overcome this challenge, we adapt the Gumbel-SoftMax reparameterization trick [17], approximating the discrete binary decision using a differentiable Gumbel-Sigmoid relaxation during training.

Training Loss. Reducing computations via channel pruning while preserving accuracy requires employing a loss function that effectively combines these goals. To preserve accuracy, we utilize the commonly used cross entropy loss, denoted as $\mathcal{L}_{task}$. The computational budget is regulated in a manner similar to the approach presented in [33]. We denote the total number of FLOPs used across all frames in the static model as $\mathbb{F}^s = \sum\sum \mathcal{F}^s_{t,\ell}$, where $\mathcal{F}^s_{t,\ell}$ is the number of FLOPs in layer l of frame t. Similarly, we count the number of FLOPs in the dynamic model, $\mathbb{F}^d = \sum\sum \mathcal{F}^d_{t,\ell}$. FLOPs are avoided at each convolutional layer for which the policy is applied to, but the obtained smaller output tensor allows further FLOPs reduction being a smaller input to the following layer. By specifying the target FLOPs density, $\mathcal{D}$, we can exert control over the computational budget of the network using a sparsity loss, formulated as the L2 norm of the difference:

$$\mathcal{L}_{spar} = [\max(\mathbb{F}^d/\mathbb{F}^s - \mathcal{D}, 0)]^2 \tag{2}$$

Combining the loss terms with a tuning coefficient λ, the overall training loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{task} + \lambda \cdot \mathcal{L}_{spar} \tag{3}$$

Policy Network Computational Overhead. In terms of FLOPs, the policy network overhead is negligible, amounting for less than 1% of the static network FLOPs (to be exact, 0.31% for ResNet18 and 0.82% for ResNet50). With respect to the number of parameters, our policy network is comparable or smaller than other policy network approaches like AdaFuse [24] and D-STEP [25].

4 Experiments

We evaluate our method on widely used datasets, namely Something-Something-V2 (194k/25k clips, 174 classes) [13], ActivityNet1.3 (10k/5k clips, 200 human activities classes) [2], Mini-Kinetics (121k/10k clips, 200 human action classes) [36], and Jester (119k/15k clips, 27 hand gesture classes) [22], all of which have predefined train/validation splits. Following [40], we distinguish between scene-focused datasets (ActivityNet1.3, Mini-Kinetics) and motion-focused datasets (Something-V2, Jester). While in scene-focused datasets videos can often be recognized by static scene appearance alone, motion-focused datasets require strong motion reasoning and the background scene contributes little to the final class prediction.

To ensure a fair comparison, we adopt the training procedures outlined in previous works, such as TSM [20]. Specifically, we uniformly sample $T = 8$ frames from each video, with each frame having an input dimension of 224×224. During training, we apply standard augmentations such as random scaling and cropping, while during inference, we employ center cropping. For network initialization, we utilize ImageNet pre-trained weights and the hyperparameters are set as follows: 50 epochs of training, an initial learning rate of 0.01, decaying by 0.1 after 20 and 40 epochs, a batch size of 64 and a weight decay of 0.0001. We set λ to 0 for 10 epochs and linearly increase it to 5 during the next 20 epochs.

Comparison with State-of-the-Art Efficient Methods. We compare our method with several leading efficient action recognition baselines: AdaFrame [38], ListenToLook [8], VideoIQ [39] SCSampler [19], FrameExit [11], OCSampler [21], AdaFuse [24], LiteEval [37] and AR-Net [23]. All of these methods are using TSN [34] as a static baseline architecture when applied on ActivityNet1.3 and add temporal shift [20] when applied on Mini-Kinetics.

We compare ourselves on ActivityNet1.3 and Mini-Kinetics datasets, in both offline and online modes. As shown in Table 1, our approach can significantly reduce computational complexity while maintaining accuracy compared to the static methods. Furthermore, our method provides excellent accuracy-efficiency trade-off compared to leading dynamic methods and reaches state-of-the-art results on the Mini-Kinetics dataset. In addition, TAPS retains a similar performance in online mode, further improving upon other methods, substantiating the advantageous TAM training in offline mode.

	Method	Backbone	ActivityNet1.3 mAP	ActivityNet1.3 GFLOPs	Mini-Kinetics Top-1	Mini-Kinetics GFLOPs
Offline	TSN (static)	ResNet50	74.9†	32.9†	–	–
	TSM (static)	ResNet50	–	–	75.2†	32.9†
	AdaFrame	MobileNetV2+ResNet101	71.5	79.0	–	–
	ListenToLook	MobileNetV2+ResNet152	72.3	81.4	–	–
	VideoIQ	ResNet50	74.8	28.1	72.3	20.4
	SCSampler	MobileNetV2+ResNet50	72.9	42.0	70.8	42.0
	FrameExit	ResNet50	76.1	26.1	72.8	19.7
	OCSampler	MobileNetV2+ResNet50	**77.2**	25.8	73.7	21.6
	AdaFuse	ResNet50	–	–	72.3	23.0
	TAPS (ours)	ResNet50	75.8	26.8	**74.9**	21.4
Online	TSN (static)	ResNet50	74.9†	32.9†	–	–
	LiteEval	MobileNetV2+ResNet101	72.7	95.1	61.0	99.0
	AR-Net	MobileNetV2+ResNet18/32/50	73.8	33.5	71.7	32.0
	FrameExit (online)	ResNet50	73.7	27.6	–	–
	TAPS (ours)	ResNet50	**75.6**	26.8	**75.2**	24.6

Table 1: Comparison with state-of-the-art dynamic methods on ActivityNet-v1.3 and Mini-Kinetics. Results are shown for both offline and online methods. † indicates results we obtained using the author's provided code.

Comparison with Dynamic Channel Pruning Methods. We compare TAPS to leading dynamic channel pruning methods, AdaFuse [24] and D-STEP [25], which apply dynamic channel pruning to video and incorporate temporal reuse by analyzing two consecutive frames. As depicted in Table 2, our method demonstrates substantial improvements across datasets and backbones. Notably, these results demonstrate the applicability of our approach to motion-focused datasets in addition to scene-focused datasets (i.e., Table 1). We provide further qualitative results in Figure 3.

Method	ResNet18				ResNet50			
	Something-V2		Jester		Something-V2		Jester	
	Top-1	FLOPs	Top-1	FLOPs	Top-1	FLOPs	Top-1	FLOPs
TSM (static)	52.6	14.6G	95.4	14.6G	59.1	33.2G	96.0	33.2G
AdaFuse	52.1	11.9G	94.7	7.9G	59.8	31.3G	95.0	21.6G
D-STEP	52.7	12.2G	95.4	7.9G	57.4	**21.4G**	95.7	16.1G
TAPS (ours)	**55.0**	**11.8G**	**95.7**	**7.5G**	**59.9**	28.7G	**96.3**	**16.1G**
AdaFuse	50.3	6.5G	90.6	**3.0G**	58.3	19.5G	94.7	16.1G
D-STEP	51.4	8.1G	94.6	3.6G	56.6	18.0G	95.2	**9.8G**
TAPS (ours)	**53.7**	**5.5G**	**95.3**	3.4G	**59.2**	**17.5G**	**95.5**	13.9G

Table 2: Comparison with Dynamic Channel Pruning methods for action recognition. Results are shown for ResNet18 and ResNet50 on mainstream datasets, across two computational budgets. Compared to TSM, TAPS saves up to 70% of FLOPs without accuracy loss, outperforming other methods by a substantial margin.

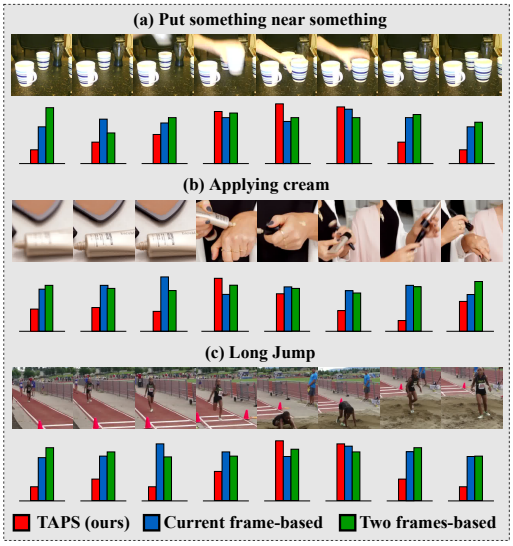

Fig. 3: Qualitative comparison between different channel pruning methods on video clips taken from Something-V2 and Mini-Kinetics. Bar plots show FLOPs per frame relative to static TSM for GAP-based policy networks using single-frame, two-consecutive frames, and TAPS.

While the majority of our experiments were carried on ResNet18/50 backbones, Table 3 compares MobileNet-V2 backbone based TAPS with its static baseline, TSM, and with current frame-based dynamic pruning. These results underscore the method's effectiveness even when applied to compact architectures.

Method	Something-V2 Top-1	Something-V2 FLOPs	Mini-Kinetics Top-1	Mini-Kinetics FLOPs
TSM (static)	54.1	2.5G	65.0	2.5G
CFP	51.8	2.4G	64.3	2.2G
TAPS (ours)	**52.2**	2.4G	**65.4**	**2.0G**
CFP	50.9	1.8G	63.9	1.8G
TAPS (ours)	**51.8**	1.8G	**64.1**	**1.7G**

Table 3: Accuracy vs FLOPs using the MobileNetV2-based TSM model in offline mode. This comparison includes TSM with TAPS and current frame-based pruning (CFP) across two computational budgets.

TAM Architecture. Temporal information may be aggregated using various methods, such as LSTM [16] and GRU [4]. In Figure 4 we compare different policy network architectures, showing that an architecture of two consecutive attention layers performs best.

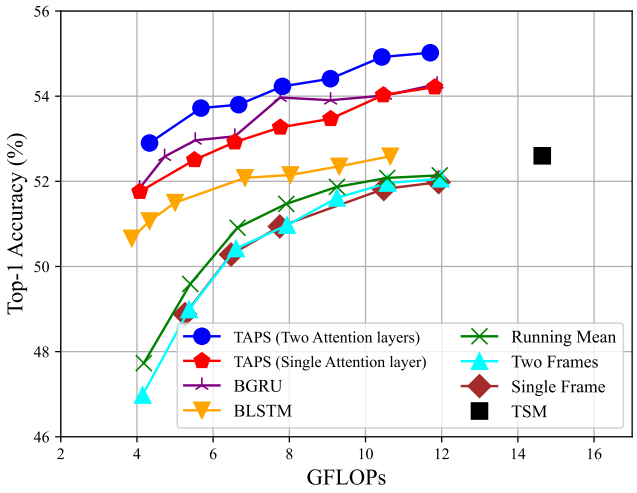

Fig. 4: Ablation on temporal aggregation modules within the policy network on Something-V2 dataset based on ResNet18 in offline mode. We compare the static TSM baseline to dynamic models using: only current frame GAP descriptor, GAP vectors from current and previous frame, running mean over GAP vectors, bidirectional LSTM, bidirectional GRU, a single attention layer and two attention layers.

Early Exit. TAPS can be combined with many other optimization approaches, static or dynamic. To demonstrate it we added an early exit operator to various online mode TAPS models. After processing each frame, we test whether the prediction logits' top score exceeds a certain naive constant threshold and avoid processing the rest of the frames accordingly. Figure 5 shows that this simple method spared 10-20% of the FLOPs, on top of TAPS savings, depending on the pruning density, with negligible degradation in accuracy and without need for additional training. This suggests that our method is orthogonal to other methods and can be successfully combined with them. We applied early exit as it was added on top of other leading methods [35], but we consider it as just one of many possible orthogonal optimizations.

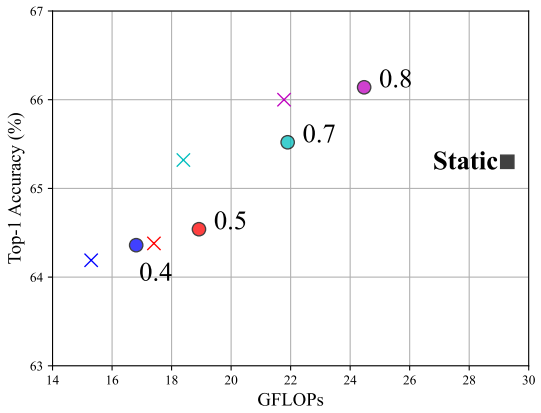

Fig. 5: Early Exit Evaluation on ActivityNet1.3 Dataset using ResNet18. Circles represent pruned models with different density targets. ×-s denote corresponding results of early exit with a threshold of 0.8 applied on the prediction logits.

Run-time Measurements. To evaluate the latency savings achieved by our method we compared a static TSM model with its dynamic versions obtained by TAPS at different target pruning rates. We simulated the dynamic network run-time in a two stage process. First, we recorded the pruning decisions of the policy networks. Then, we created static networks by removing the pruned filters from the original network, and measured their GPU run-time. The static ResNet50 TSM model exhibited a mean run-time of 10.75 milliseconds per clip on Something-V2 (with a batch size of 12) on a NVIDIA 1080Ti GPU. Dynamic versions with 13%, 35%, and 50% less FLOPs achieved run-time reductions of 9.2%, 23.2%, and 29.7%, respectively. These results demonstrate that the FLOPs reduction presented above translates to significant GPU latency reduction, affirming the practicality of our approach.

Scaling Mask Effect. We assess the contribution of temporal scaling through two experiments. We first concentrate on static models and compare scaling based on temporal aggregation (TAS) to scaling based solely on the current frame (CFS) and no scaling at all. We apply these configurations on TSM with a varying number of frames to obtain different working points. The second experiment compares current frame-based pruning and temporal aggregation-based pruning, with (TAPS) and without (TAP) scaling, at different density targets. Figure 6 shows that for both experiments the effect of temporal aggregation-based scaling is pronounced, exhibiting an approximate gain of 1% in performance across all configurations.

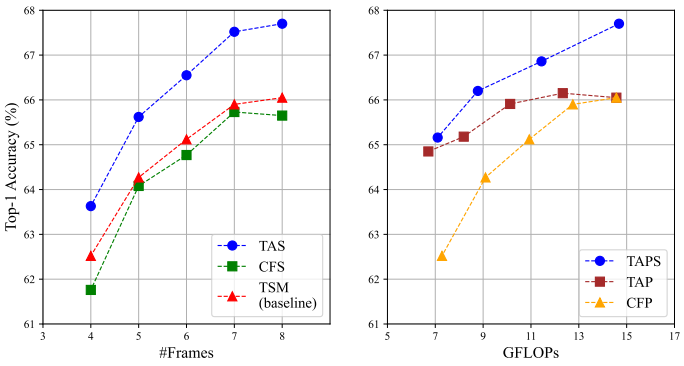

Fig. 6: The Effect of Scaling demonstrated on Mini-Kinetics dataset using ResNet18: (Left) Static models (i.e., without pruning) with varying numbers of sampled frames: temporal aggregation-based scaling (TAS) compared to current frame-based scaling (CFS) and TSM baseline. **(Right)** Dynamic models with different density targets, with (TAPS) and without (TAP) scaling, compared to current frame-based pruning (CFP).

Channel-wise and Layer-wise Statistics We provide pruning statistics of models obtained by applying TAPS on ResNet18 trained on Something-Something-V2. For each model, we calculate the *compute rate per channel*, i.e., the percentage of frames in which each channel was computed, averaged over all video clips. As shown in Figure 7, TAPS performs dynamic pruning, taking advantage of the content-dependent pruning mechanism, rather than converging to a content-agnostic pruning. This supports the applicability of a dynamic policy network compared to static pruning methods. The statistics also show that the deeper the layer, the more dynamically its channels behave. For example, for ResNet18 with a density target of 0.8, the amount of channels that are constantly retained (not pruned for any frame/video) decreases as the layer depth increases. For ResNet18

with a density target of 0.3, most of the channels in the first layer are static while most of the channels in the last layer are not. This observation is aligned with the typical features extracted at each layer; shallow layers often extract low-level features (e.g., edges) which are content-agnostic whereas deep layers often extract high-level features (e.g., cats).

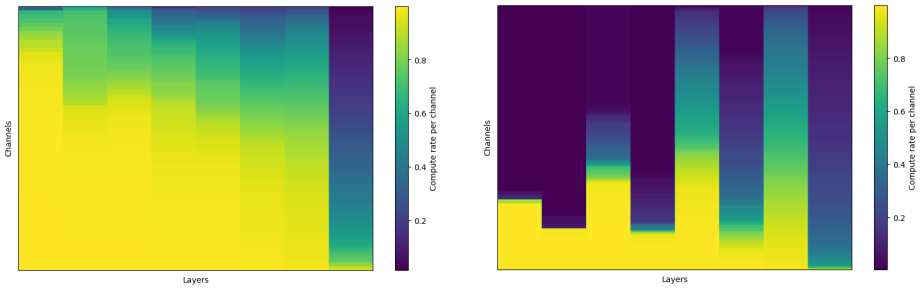

Fig. 7: Pruning statistics of models obtained by applying TAPS on ResNet18 trained on Something-V2, with a density target of 0.8 (left) and 0.3 (right) (11.8 GFLOPs and 5.5 GFLOPs, respectively). Layers are ordered left-to-right according to their depth. In each layer, channels are sorted according to compute rate.

Pruning Variation and Content-Dependency We provide additional qualitative results based on TAPS applied on ResNet50 on Mini-Kinetics and SomethingV2 in Figure 8. For each dataset, we present two video samples from the same class, one that resulted with a higher pruning rate and one that resulted in a lower pruning rate. As shown in Figure 8 (left), on Mini-Kinetics, which is a scene-focused dataset, TAPS is able to reduce computations for videos with clear appearances while requiring more computations for videos with low resolution, hard lighting conditions or non-standard pose. Figure 8 (right) shows that on SomethingV2, which is a motion-focused dataset, TAPS incurs more computations for videos with more motion.

5 Conclusion

This paper proposes a dynamic approach for reducing the computational load of neural networks for video processing. TAPS utilizes lightweight policy networks to make channel-wise pruning decisions based on, for the first time, features accumulated from all frames. TAPS is also the first method to use temporal attention within the policy networks, and proposes a generalization of the hard pruning decision in the form of soft channel scaling. TAPS applies to a variety of video architectures, and presents a new state-of-the-art in terms of the accuracy-efficiency trade-off for the action recognition task.

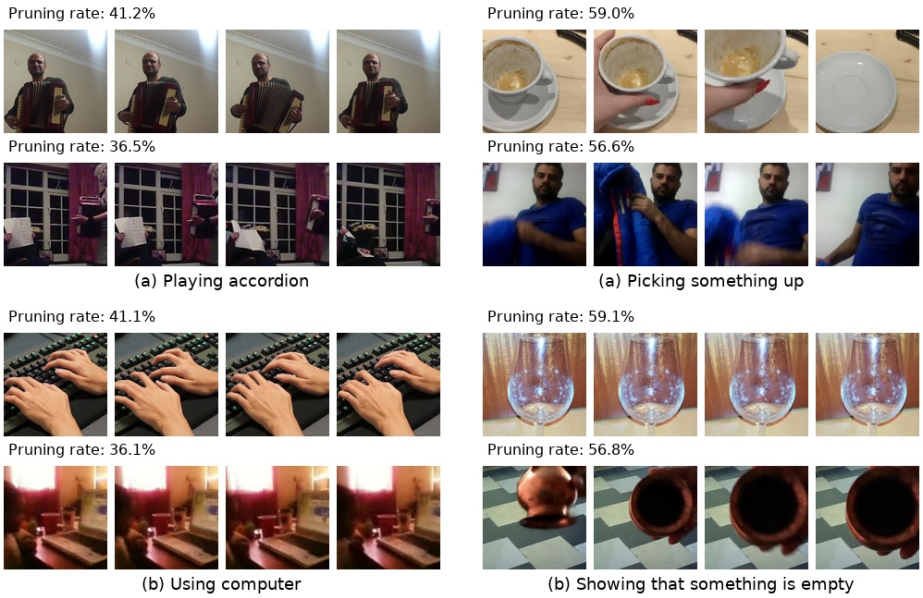

Fig. 8: Sample video clips from Mini-Kinetics (left) and Something-V2 (right) resulting in higher and lower pruning rates, for two different classes.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6836–6846 (October 2021)
2. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: 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)
3. Carreira, J., 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)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR **abs/1412.3555** (2014), `http://arxiv.org/abs/1412.3555`
5. Dai, R., Das, S., Kahatapitiya, K., Ryoo, M.S., Brémond, F.: Ms-tct: multi-scale temporal convtransformer for action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20041–20051 (2022)
6. Dror, A.B., Zehngut, N., Raviv, A., Artyomov, E., Vitek, R.: Layer folding: Neural network depth reduction using activation linearization. In: 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022. BMVA Press (2022), `https://bmvc2022.mpi-inf.mpg.de/0612.pdf`

7. Feichtenhofer, C.: X3d: Expanding architectures for efficient video recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 203–213 (2020)
8. Gao, R., Oh, T.H., Grauman, K., Torresani, L.: Listen to look: Action recognition by previewing audio (2020)
9. Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. arXiv preprint arXiv:1810.05331 (2018)
10. Gao, Y., Zhang, B., Qi, X., So, H.K.H.: Dpacs: Hardware accelerated dynamic neural network pruning through algorithm-architecture co-design. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. pp. 237–251 (2023)
11. Ghodrati, A., Bejnordi, B.E., Habibian, A.: Frameexit: Conditional early exiting for efficient video recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
12. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 244–253 (2019)
13. Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al.: The "something something" video database for learning and evaluating visual common sense. CoRR **abs/1706.04261** (2017), `http://arxiv.org/abs/1706.04261`
14. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(11), 7436–7456 (2021)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
17. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
18. Koot, R., Lu, H.: Videolightformer: Lightweight action recognition using transformers (2021), `https://arxiv.org/abs/2107.00451`
19. Korbar, B., Tran, D., Torresani, L.: Scsampler: Sampling salient clips from video for efficient action recognition (2019)
20. Lin, J.: Tsm: Temporal shift module for efficient video understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7083–7093 (2019)
21. Lin, J., Duan, H., Chen, K., Lin, D., Wang, L.: Ocsampler: Compressing videos to one clip with single-step sampling (2022)
22. Materzynska, J., Berger, G., Bax, I., Memisevic, R.: The jester dataset: A large-scale video dataset of human gestures. In: Proceedings of the IEEE/CVF international conference on computer vision workshops (2019)
23. Meng, Y., Lin, C.C., Panda, R., Sattigeri, P., Karlinsky, L., Oliva, A., Saenko, K., Feris, R.: Ar-net: Adaptive frame resolution for efficient action recognition. In: European Conference on Computer Vision. pp. 86–104. Springer (2020)
24. Meng, Y., Panda, R., Lin, C.C., Sattigeri, P., Karlinsky, L., Saenko, K., Oliva, A., Feris, R.: Adafuse: Adaptive temporal fusion network for efficient action recognition. arXiv preprint arXiv:2102.05775 (2021)

25. Raviv, A., Dinai, Y., Drozdov, I., Zehngut, N., Goldin, I.: D-step: Dynamic spatio-temporal pruning. In: 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022. BMVA Press (2022), https://bmvc2022.mpi-inf.mpg.de/0632.pdf
26. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
27. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. Advances in neural information processing systems **27** (2014)
28. Tian, Y., Lu, G., Yan, Y., Zhai, G., Chen, L., Gao, Z.: A coding framework and benchmark towards low-bitrate video understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
29. Tian, Y., Lu, G., Zhai, G., Gao, Z.: Non-semantics suppressed mask learning for unsupervised video semantic compression. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13610–13622 (2023)
30. Tian, Y., Yan, Y., Zhai, G., Guo, G., Gao, Z.: Ean: event adaptive network for enhanced action recognition. International Journal of Computer Vision **130**(10), 2453–2471 (2022)
31. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (December 2015)
32. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., Baik, S.W.: Action recognition in video sequences using deep bi-directional lstm with cnn features. IEEE access **6**, 1155–1166 (2017)
33. Verelst, T., Tuytelaars, T.: Dynamic convolutions: Exploiting spatial sparsity for faster inference. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2320–2329 (2020)
34. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Gool, L.V.: Temporal segment networks: Towards good practices for deep action recognition. In: European conference on computer vision. pp. 20–36. Springer (2016)
35. Wang, Y., Yue, Y., Lin, Y., Jiang, H., Lai, Z., Kulikov, V., Orlov, N., Shi, H., Huang, G.: Adafocus v2: End-to-end training of spatial dynamic networks for video recognition. arXiv preprint arXiv:2112.14238 (2021)
36. Will, Kay, J.C., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. CoRR **abs/1705.06950** (2017), http://arxiv.org/abs/1705.06950
37. Wu, Z., Xiong, C., Jiang, Y.G., Davis, L.S.: Liteeval: A coarse-to-fine framework for resource efficient video recognition (2019)
38. Wu, Z., Xiong, C., Ma, C.Y., Socher, R., Davis, L.S.: Adaframe: Adaptive frame selection for fast video recognition. CoRR **abs/1811.12432** (2018), http://arxiv.org/abs/1811.12432
39. Zhang, Y., Bai, Y., Wang, H., Xu, Y., Fu, Y.: Look more but care less in video recognition. arXiv preprint arXiv:2211.09992 (2022)
40. Zhu, Y., Li, X., Liu, C., Zolfaghari, M., Xiong, Y., Wu, C., Zhang, Z., Tighe, J., Manmatha, R., Li, M.: A comprehensive study of deep video action recognition. arXiv preprint arXiv:2012.06567 (2020)