Selective Feature Compression for Efficient Activity Recognition Inference

Chunhui Liu*[†], Xinyu Li*, Hao Chen, Davide Modolo, Joseph Tighe
Amazon Web Services
{chunhliu, xxnl, hxen, dmodolo, tighej}@amazon.com

Abstract

Most action recognition solutions rely on dense sampling to precisely cover the informative temporal clip. Extensively searching temporal region is expensive for a real-world application. In this work, we focus on improving the inference efficiency of current action recognition backbones on trimmed videos, and illustrate that an action model can accurately classify an action with a single pass over the video unlike the multi-clip sampling common with SOTA by learning to drop non-informative features. We present Selective Feature Compression (SFC), an action recognition inference strategy that greatly increases model inference efficiency without compromising accuracy. Different from previous works that compress kernel size and decrease the channel dimension, we propose to compress features along the spatio-temporal dimensions without the need to change backbone parameters. Our experiments on Kinetics-400, UCF101 and ActivityNet show that SFC is able to reduce inference speed by 6-7x and memory usage by 5-6x compared with the commonly used 30 crop dense sampling procedure, while also slightly improving Top1 Accuracy. We perform thorough quantitative and qualitative evaluation and show how our SFC learns to attend to important video regions for the task of action recognition.

1. Introduction

Action recognition with 3D CNNs has seen significant advances in recent years [2, 4, 5, 18, 20, 22–24, 28–30], thanks to their ability to implicitly model motion information along with the semantic signals. However, these popular 3D models require a substantial amount of GPU memory and therefore cannot operate on long video sequences. Instead, they sample short video clips of 0.2-4 seconds and independently classify action labels for each clip. Finally, they pool the results from all clips together to generate a final video-level prediction. How to sample these clips plays a critical role in these models and several techniques have been proposed (fig. 1, table 1). Among these, *dense sampling* is considered the best solution and employed in almost all action recognition models. Dense sampling works in a temporal sliding window manner and it over-samples overlapping segments exhaustively, ensuring that no information is missed and that the action is precisely localized within a window. While this achieves the best performance, it is also highly inefficient, as a lot of the sampled clips are highly related to each other (redundant) or not informative for the classification prediction of the video (irrelevant). Recently, SCSampler [12] and MARL [29] proposed to learn to select a small subset of relevant clips to reduce this dense prediction problem. While they reach very promising results, their

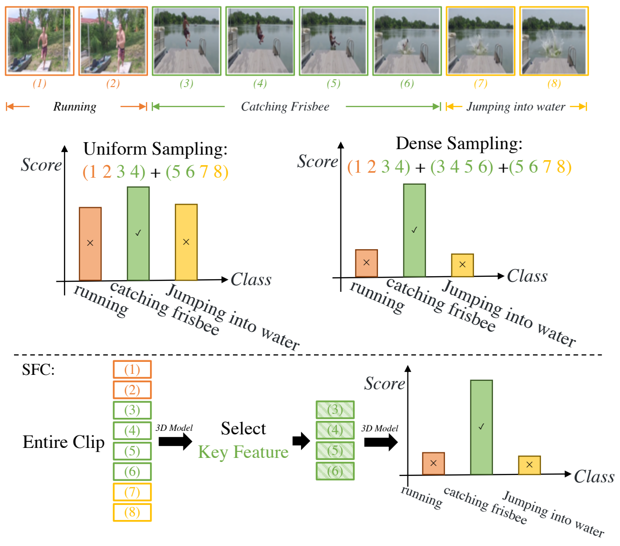

Figure 1: *With fast motion, uniform sampling (row 1, left) is not guaranteed to precisely locate the key region. Consequently, an action model that uses it can be distracted by noise and can fail to recognize the correct action. Instead, dense sampling (row 1, right) looks at all the possible regions and can precisely locate the action and ignore noisy regions via voting. While accurate, dense sampling is however very inefficient. We propose SFC (row 2) to avoid sampling and instead it looks at the whole video and compresses its features into a representation that is smaller and adequate for action recognition.*

*Equal contribution
[†]Work done at Amazon, Currently at Kuaishou Technology

hard selection mechanism can potentially disregard useful information, especially on videos with complex actions.

In this paper we propose a novel technique that removes the need for dense sampling via feature compression. We call it *Selective Feature Compression (SFC)*. SFC looks at long video sequences (up to two minutes) and compresses the rich information of their frames into a much smaller representation, which is then analyzed by a more expensive 3D network that predicts the corresponding video action labels (fig. 1). This method achieves a significant improvement in inference speed, without a drop in accuracy. In detail, we propose to split a pre-trained action network into two sub-networks (head and tail) and place our SFC in between them. Our SFC compresses the internal spatial-temporal representations from the head network along the temporal dimension using an attention-based mechanism that learns global feature correlation within the entire video sequence. This compressed representation is passed to the tail network, which now operates on a much smaller input. This design brings the additional benefit of not needing to retrain the action network, as SFC can be finetuned independently and very quickly. Furthermore, SFC offers a good trade-off between inference and performance that can be easily tuned based on necessity (e.g., for fast real-world applications we can compress more aggressively at the cost of some performance).

To validate the effectiveness of SFC, we present an extensive analysis on the popular Kinetics 400 dataset [2] and UCF101 [21] datasets. Our results show that SFC works with a wide range of backbones and different pre-trainings. SFC maintains the same top-1 accuracy of dense sampling (30 crops), while improving the inference throughput by 6-7$\times$ and reducing the memory usage by 6$\times$. While we designed SFC to replace the dense sampling strategy for action recognition on short trimmed videos, we also investigate its applicability on longer untrimmed content. We present results on ActivityNet [3] datasets, where we show that SFC can be used in conjunction with uniform sampling and improve both performance and runtime. Finally, we present a visual analysis showing how SFC focuses on the informative parts of a video during feature compression.

2. Related Work

As introduced in the previous section, sampling plays a critical role in action recognition (table 1). Early video classification models were trained on individual frames, as opposed to video clips. *Sparsely sampling frames* [10] uniformly or within segment-based clips [27] were popularly used. Although efficient, sparse frame sampling does not work well with 3D networks [2, 20, 22–24, 30], as it breaks the temporally continuity required by these 3D models. The most intuitive way to run video inference with 3D networks is to take the entire video as input and run *fully convolutional inference* [8, 16, 26, 32]. However, the fully convolutional inference does not scale well for long videos as the memory requirements are well above the capabilities of modern GPUs [5, 28]. Inspired by image classification, multi-crop *dense sampling* inference [28] improved accuracy and reduced memory need per iteration, by performing fully convolutional inference on a sampled set of video crops. While this procedure remains one of the most widely used [4, 5, 13, 14, 17, 28, 31], the memory consumption and computation complexity in video level are instead greatly increased.

Method	Accuracy	Memory	Latency	Generalization
Sparse Sampling	Medium	Low	Low	High
Fully Convolutional	Medium	Medium	Low	High
Dense Sampling	High	High	High	High
Kernel Compression	Medium	Medium	Medium	Low
Input Sampling	High	Medium	Medium	High
This paper: SFC	High	Low	Low	High

Table 1: *Comparison of different inference strategies. Our SFC module tries to improve action recognition inference along all these aspects.*

While the research on the topic of efficient inference remains limited, it is starting to gain more attention. One typical inference speed-up strategy is using *kernel compression* to reduce the total computation required for one pass of the network [11, 18]. Some other methods reduce the computation by distilling the efficient convolution kernel [34] or use 2D convolution for spatial-temporal modeling by temporal feature shifting [1, 15]. However, these special customized modules need fully re-training the backbone weights, making them rather difficult to generalize and to benefit from recent progress on model design and data. Differently from these methods, SFC does not require manipulating the backbone network weights, and it is much simpler. Moreover, note that kernel compression methods still employ the heavy multi-crop inference and they can benefit from this paper's new way of doing a single pass inference using SFC.

Following the idea of input sampling used for other video tasks [7, 9, 19, 33], another way to boost inference speed is through applying strategically *input sampling* on the input video. For example, [29] uses a light weight network to sample a few frames and [12] uses it to sample subclips, achieving about 2$\times$ speed-up and maintaining accuracy. However, those methods involve a two-stage training which includes learning a selection network by reinforcement learning or oracle loss design. The added selection network brings additional computation, which limits the efficiency and memory improvement. Our SFC shares the same core idea of improving inference by selecting relevant video information, but it does so at the feature level.

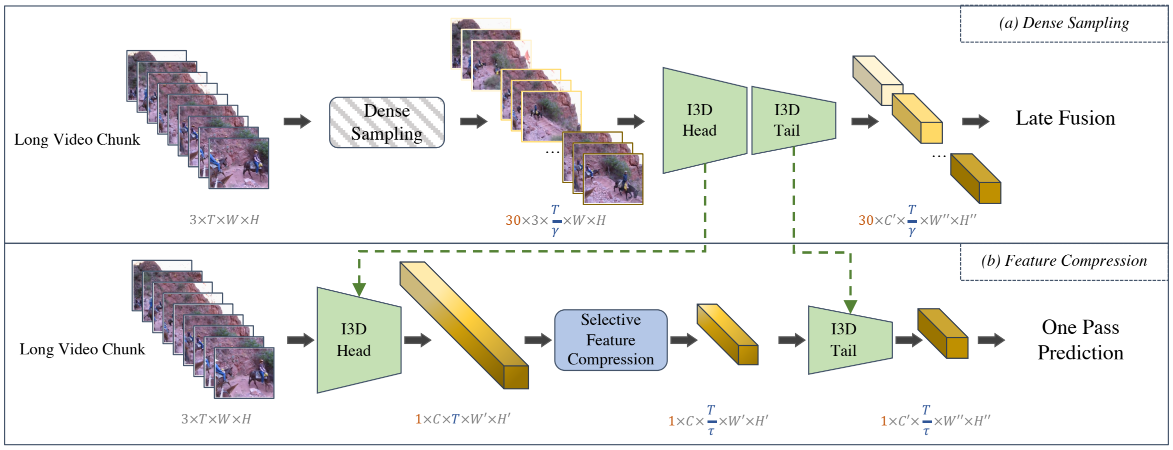

Figure 2: *Dense sampling inference v.s. our SFC procedure. Instead of running inference on multiple crops and (late) fusing their predictions, SFC runs inference once on the whole video sequence. Moreover, the SFC is placed within the action recognition network, so that it can compress the output of I3D Head and reduce inference time of I3D Tail.*

3. Learn to look by feature compression

Most state-of-the-art action recognition approaches [2, 20, 22–24, 30] train on short video crops randomly sampled from long training videos. At inference, they densely sample a test video, dividing it into a number of crops (i.e., 30 [4, 5, 28] and 10 [12] for Kinetics), run inference on each of these independently, and average their predictions into a video-level output (Fig. 2*top*). While these methods have successfully achieved outstanding action recognition performance, they lack inference efficiency. Instead, we propose an efficient solution that removes the need for dense sampling, while also reducing the inference time of the 3D backbone by compressing its features (fig. 2*bottom*).

Our method works by taking a pre-trained 3D network encoder and splitting it into two components, *head* and *tail*. We then insert our novel Selective Feature Compression module between the *head* and *tail* networks. We design our SFC to: (i) remove redundant information while preserving useful parts of the video directly at the feature level and (ii) reduce the inference computation of the *tail* network by compressing the feature.

Formally, given an input video V, our approach predicts activity labels a as follows:

$$a = \Theta_{tail}(\Phi(\Theta_{head}(V))), \quad (1)$$

where Θ_{head} and Θ_{tail} represent the decoupled head and tail components and Φ represents our SFC. The feature compression operation is learnt by a cross entropy loss using original action labels Y:

$$\min_{\Phi} \ \mathcal{L}_{\text{CE}}\left(Y; \Theta_{tail}(\Phi(\Theta_{head}(V))\right) \quad (2)$$

We present the design for our SFC module in the next section (Sec. 3.1).

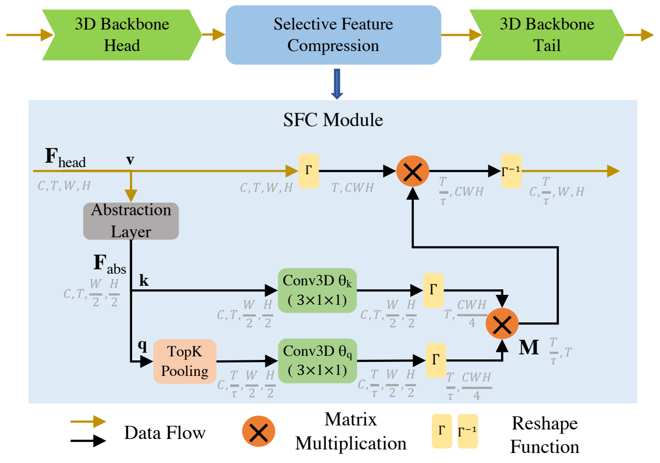

Figure 3: *Selective Feature Compression design. Our proposed design is capable of modelling long-distance interactions among features and better discover the most informative video regions.*

3.1. Selective Feature Compression

Our solution is inspired by the self-attention mechanism [25, 28], as it can directly model long-distance interactions between features to automatically discover what to attend to, or in our case, what to compress. However, differently from the original formulation of self-attention that takes an input sequence of length n and outputs a sequence of the the same length [28], we want SFC to return a shorter (compressed) sequence of length n/τ that is compatible with the frozen *tail*. To satisfy these requirements, we model SFC as shown in Fig. 3. Formally, given the feature generated by the *head* sub-network $\mathbf{F}_{head} = \Theta_{head}(V) \in \mathbb{R}^{C \times T \times W \times H}$, we formulate SFC as:

$$\Phi = \text{SFC}(\mathbf{k} = \mathbf{q} = \mathbf{F}_{abs}, \mathbf{v} = \mathbf{F}_{head}) \quad (3)$$
$$= \mathbf{M} \cdot \mathbf{v}, \quad (4)$$
$$\mathbf{M} = \text{softmax}(\theta_q(\text{pool}(\mathbf{q}))^T \cdot \theta_k(\mathbf{k})), \quad (5)$$

where θ_k and θ_q are linear transformations implemented as $(3 \times 1 \times 1)$ 3D convolution kernels, pool refers to our TopK Pooling module, $\mathbf{F}_{abs} = \Theta_{abs}(\mathbf{F}_{head})$ are the head features re-encoded with our abstraction network and $\mathbf{M}$ is the attention map (note: we omit the reshape function Γ in the equation for simplicity, but include it in Fig. 3).

Although this looks similar to a classic self-attention/non-local block, as $\mathbf{A}(\mathbf{k} = \mathbf{q} = \mathbf{v} = \mathbf{F}_{head}) = \mathbf{M} \cdot \theta_v(\mathbf{v})$, with $\mathbf{M} = \text{softmax}(\theta_q(\mathbf{q})^T \cdot \theta_k(\mathbf{k}))$, our SFC is designed for a different purpose, and differs in the following key ways that are critical to our methods effectiveness. We present ablation studies in Section 5 to show that those details are crucial for a high performance.

Abstraction layer Θ_{abs}. Θ_{head} is trained for action classification and therefore, $\mathbf{F}_{head}$ captures information that is important for that task, like motion patters and object/place semantics. However, we argue that this information is not optimal for feature compression and thus we introduce an abstraction module that re-encodes $\mathbf{q}$ and $\mathbf{k}$ using two ResNet Blocks. This transforms the features from low-level motion patterns to more meaningful representations for compression ($\mathbf{F}_{abs}$). Finally, note that we only re-encode $\mathbf{k}$ and $\mathbf{q}$ using this layer, as we want to (i) specialize it for compression and (ii) preserve the features of $\mathbf{v}$ for compatibility with Θ_{tail}.

TopK Pooling. We use pooling to downsample the features of the query $\mathbf{q}$ from T to T/τ in the temporal dimension. This ensures that the output of SFC is a compressed vector. Instead of using average/max pooling which compress locally within a small window, we propose to use TopK pooling to allow the compression to select features that are temporally consecutive. Given the feature $\mathbf{F}_{abs} \in \mathbb{R}^{C \times T \times W \times H}$, TopK pooling (with downsample ratio $(\tau, 1, 1)$) returns $\mathbf{F}_{pool} \in \mathbb{R}^{C \times \frac{T}{\tau} \times W \times H}$ that contains the top T/τ highest activated features along time T.

Value $\mathbf{v}$. In addition to not re-encoding $\mathbf{v}$ using Θ_{abs}, we also avoid transforming $\mathbf{v}$ with θ_v, as this could also break compatibility with the Θ_{head}. Instead, we train SFC to directly attend to $\mathbf{v}$ using the attention map $\mathbf{M}$ (eq. 4).

4. Experimental Settings

Backbones. To test the generalization ability of SFC, we plug it into some of the most popular activity recognition backbones: Slow-Only I3D-50 [5], TPN-50 [31], and R(2+1)D-152 [24]. For Slow-Only I3D, we experiment with three sample rates: 2, 4, and 8 with input length equal to 32, 16, and 8 respectively. For Slow-Only I3D and TPN, we use the publicly pre-trained models released in [31], where Slow-Only 16×4 works slightly better than 32×2. For R(2+1)D-152, we use the model pre-trained on IG-65M and released in [6].
Baseline. For dense sampling, we follow the literature [5, 28] that uniformly crops 10 clips temporally and 3 clips spatially, for a total of 30 crops. Please note that numbers for Slow-Only network are slightly different from [5]. This is caused by slight variations in the videos of the Kinetics dataset, as some have been removed from YouTube.

Implementation details. We start with a pre-trained 3D backbone and divide it into *head* and *tail* (using residual block 3 as the cutting point). We freeze head and tail weights (including BN) and insert our SFC between them.

During ***training***, we generate gradients for SFC and *tail*, but only update the weights of SFC. We use the same augmentation and hyper-parameters that are used to train the original backbone, but pass the whole video as input instead of short clips. For training, we use 32 Tesla V100 GPUs for 15 epochs only, as SFC is light-weight and converges quickly. We set the initial learning rate to 0.01 and drop it by a factor of 10 at epoch 8 and again at epoch 12. We use an SGD optimizer with the weight decay set to 1e-5 and momentum to 0.9. To avoid over-fitting, we apply several data augmentations, like outward scaling, cropping (224×273) and horizontal flipping. During ***inference***, instead, we resize each video to 256×312.

Datasets. We present results on three popular action recognition datasets: Kinetics 400 [2], UCF101 [21] and ActivityNet [3]. Kinetics 400 consists of approximately 240k training and 20k validation videos trimmed to 10 seconds and annotated with 400 human action categories. UCF101 is a smaller dataset with 13k videos annotated with 101 action categories. ActivityNet (v1.3) [3] is an untrimmed dataset consisting of 19k videos, many of which are long (5 to 20 minutes), with 200 action classes. We report numbers on validation, as testing labels are not publicly available.

Evaluation Metrics. In order to fully evaluate SFC for a practical usage, we use the following metrics:

1. *Accuracy.* We report Top1 and Top5 classification accuracy to evaluate action recognition performance.
2. *FLOPS.* We report floating point operations per-second to evaluate inference runtime. Note that we compute this over a video, while many previous works

Backbone	Trained	Inference	Input	Efficiency FLOPS	Efficiency Throughput	Memory #Params.	Memory #Videos	Accuracy Top1	Accuracy Top5
Slow Only I3D-50 8×8	K400	30 crops	64×30	1643G	2.4	32.5M	3	74.4	91.4
		SFC	288×1	**247G**	**14.8 (6×)**	35.9M	**19**	**74.6**	**91.4**
Slow Only I3D-50 16×4	K400	30 crops	64×30	3285G	1.2	32.5M	1	76.4	92.3
		SFC	288×1	**494G**	**7.4 (6×)**	35.9M	**8**	**76.9**	**92.5**
Slow Only I3D-50 32×2	K400	30 crops	64×30	6570G	0.6	32.4M	<1	75.7	**92.3**
		SFC	288×1	**988G**	**3.6 (6×)**	35.9M	**4**	**75.8**	92.1
TPN-50 8×8	K400	30 crops	64×30	1977G	2.1	71.8M	2.7	76.0	**92.2**
		SFC	288×1	**359G**	**12.5 (6×)**	85.7M	**17**	**76.1**	92.1
R(2+1)D-152	IG-65M→	30 crops	64×30	9874G	0.4	118.2M	<1	79.4	94.1
	K400	SFC	288×1	**1403G**	**2.8 (7×)**	121.7M	**5**	**80.0**	**94.5**

Table 2: *Comparison of different backbones with dense sampling inference and with SFC on Kinetics 400. Differently from the other entries, R(2+1)D was initially pre-trained on the IG-65M datasets and later trained on K400. Given any backbone, we freeze it an train our SFC module only on K400 dataset. Finally, we compare the results with dense sampling (30 crops).*

τ	Data Used	FLOPs	Throughput	Top1	Top5
1	100%	352G	9.6	73.8	91.2
4/3	75%	299G	12.1	74.5	91.4
$\underline{2}$	50%	247G	14.8	**74.6**	**91.4**
4	25%	195G	17.8	72.0	90.1

Table 3: *The impact of different data using ratio to speed-accuracy tradeoff, using I3D-50 8×8.*

Method	Throughput	Top1
Baseline: 30 crops	1×	74.5
Single Crop	30×	−7.2
TSN Single Sampling [27]	30×	−5.7
TSN Dense Sampling [27]	3×	−4.9
SCSampler [12]	2×	+2.5
SFC	6.2×	+0.2

Table 4: *Comparison with input sampling methods.*

reported it over clips. For a fair comparison, clip-level FLOPS is multiplied by N when N crops are sampled for one video.

3. *Video Throughput*. As FLOPS cannot always estimate precise GPU speed due to different implementations of the model's operations, we also report video throughput speed, which is the number of videos that a single Tesla V100 GPU can process per second (excluding data loading).
4. *Number of model parameters*. We use this number to report the complexity of a model.
5. *Number of videos per batch*. In addition to model complexity, we also report the number of videos we can perform inference simultaneously (i.e., in a batch) on a single GPU (of 16GB, Tesla V100). This number show another perspective on the efficiency of an inference strategy.

5. Experimental Results

In this section we present a series of experiments. First, we show that SFC can considerably improve inference efficiency without affecting model accuracy (sec. 5.1) on two trimmed dataset: Kinetics-400 and UCF101. Then, we ablate the components of SFC to validate our design hypothesis (sec. 5.2) on Kinetics-400.

Method	FLOPS	#Params.	Top1
TSM [15]	64G × 10	24M	95.9
I3D [2]	65G × 30	44M	92.9
NonLocal R50 [28]	65G × 30	62M	94.6
Slow I3D 8×8, 1 Crop	54G ×1	32.5M	93.8
Slow I3D 8×8, 30 Crops	54G ×30	32.5M	**94.5**
Slow I3D 8×8 + **SFC**	247G	35.9M	**94.5**

Table 5: *Results on UCF 101.*

5.1. Results on Kinetics-400 and UCF101

Evaluating different backbones. We equip several popular action recognition backbones with SFC and evaluate their accuracy, inference efficiency and memory usage compared on Kinetics-400. We then compare their results to those obtained by the popular 30 crops dense sampling (table 2 figure 4). Results show that SFC generalizes well to different backbones, with different sampling rates (rows 1-3, using 8×8, 16×4 and 32×2), different backbone designs (rows 1, 4, 5, as SlowOnly, TPN, R(2+1)D) and different data pre-training (row 5, as uing IG-65M [6]). In general we observe that SFC improves the video-level inference efficiency by $6 - 7\times$ of dense sampling, without losing any performance. We argue that this is thanks to SFC 's ability to drop redundant and irrelevant information. Moreover, note how our compression design also reduces memory usage greatly, by around $7\times$. This is because SFC does not

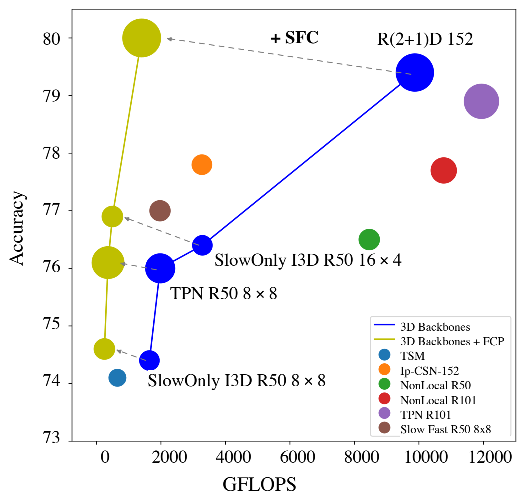

Figure 4: *Inference speed (GFLOPS) vs Performance (Top1 Accuracy) vs Model size (Number of parameters, bubble size). Solid lines link the backbones evaluated in table 2 and dashed arrows show the performance improvement between using 30 crops dense sampling (blue) and SFC (yellow) with those backbones.*

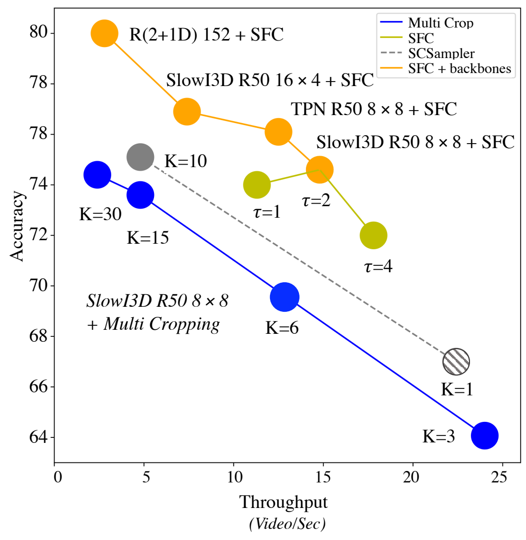

Figure 5: *Comparing speed-accuracy trade-off with different sampling methods. The blue curve represents the dense sampling method using SlowOnly I3D, with different numbers of crops (K) sampled. Results of K equaling to 30 (10 × 3), 15 (5 × 3), 6 (2 × 3) and 3 (1 × 3) are reported. The yellow curve represents our SFC method using SlowOnly I3D with different compression ratios τ, i.e., only using $\frac{1}{\tau}$information. We can observe that our proposed SFC provides a better inference solution.*

need to analyze overlapping temporal windows and it can drop non-informative temporal regions.

Results also show that a very expensive (and well-performing) R(2+1)D-152 solution can be run at the same throughput of the much cheaper I3D-50 8×8, when equipped with SFC. In practice, thanks to SFC we are able to achieve the best accuracy at the same speed of a relatively fast model. Moreover, dense sampling on R(2+1)D-152 requires 22GB of memory for a single video (assuming 30 crops), which is extremely expensive compared to the 3GB usage by SFC.

Speed-performance trade-off. To understand the trade off between the compression ratio and performance, we ablate the compression ratio in table 3 and fig. 5. Results show that some compression is actually beneficial and leads to small improvements in performance, as SFC is able to disregard useless information for classification. This was also observed in previous works [12, 29]. While compressing more aggressively (i.e., $\tau = 4$) leads to fast inference, but slightly lower performance. We find that a compression ration of 50% (i.e., $\tau = 2$), offers a good trade-off and achieves the best overall performance on K400.

Data selection. We compare our feature compression idea against data selection/sampling methods using I3D-50 8×8 (table 4). We compare against TSN [27] sampling, which proposes to temporal segment the videos and extract discontinuous frames for 3D models, and the recent SCSampler, which uses a cheap network to select what segments to run full inference on. While TSN sampling is able to improve throughput, it also degrades performance quite severally, compared to the baseline. On the other hand, SCSampler and SFC can improve both the performance and the throughput. As SCSampler performs selection rather than compression, it disregard regions more aggressively than SFC and can achieve better performance than SFC, thought SFC is 3× more efficient. Finally, we would like to note that, in theory, selection methods (like SCSampler) work well on videos with simple actions, but may not generalize as well to more complex videos with a variety of important information. This because they are forced to (hard) select a subset of segments. Our SFC compresses by looking at the whole video and thus can preserve all needed information.

Evaluating on UCF101. Finally, we present results on UCF101 in table 5. Similarly to the Kinetics-400 results, SFC is able to match the performance of dense sampling (30 crops), while being 6.5× more efficient. Overall, we believe that SFC offers a good trade-off between fast inference and competitive performance, which is one of the keys to enable action recognition for real-world applications.

5.2. Ablation Study of SFC

In this section we investigate how SFC performance and efficiency change with respect to some of its parameters. Specifically, we present an ablation study on 4 important

SFC Insertion Point	FLOPs	Throughput	Top 1	Top5
Res 1234 / Res 5	305G	14.0	73.7	90.7
Res 123 / Res 45	247G	14.8	**74.6**	**91.4**
Res 12 / Res 345	211G	15.2	73.2	90.5
Res 1 / Res 2345	163G	21.2	73.1	90.9

(a) Backbone Split Choices

pool($\mathbf{q}$)	Top1
Average Pooling	72.9
Max Pooling	72.6
TopK Pooling	**74.6**

(b) Pooling Strategy

Model	Top1
$\mathbf{k} = \mathbf{q} = \mathbf{v} = \mathbf{F}_{bead}$	72.7
$\mathbf{k} = \mathbf{q} = \mathbf{v} = \mathbf{F}_{abs}$	63.2
$\mathbf{k} = \mathbf{q} = \mathbf{F}_{abs}, \mathbf{v} = \mathbf{F}_{head}$	**74.6**

(c) KQV Choices

Model	FLOPS	#Params.	Top1
Conv (3x3x3)	356G	48.52M	72.0
Conv (3x1x1)	247G	35.93M	**74.6**
Conv (1x1x1)	238G	34.89M	72.5

(d) Different Conv Kernels

Table 6: *Ablation studies on Kinetics-400, using I3D-R50 8×8.*

components using a I3D-50 8×8 backbone:

Head/Tail Splitting Point. (table 6a). We investigate different splits of the action recognition backbone. Each option corresponds to a different location for our SFC insertion. Results show that all entries offer competitive results, with the best accuracy achieved by splitting after layer 3. Splitting after layer 1 offers the fastest throughput, at the cost of 1.5 Top1 accuracy. This is an interesting result for applications that require very fast inference.

Pooling Strategy (table 6b). We choose TopK pooling in SFC to temporally downsample the features of the query $\mathbf{q}$, because it can choose continuous temporal information and achieve the highest compatibility with *tail*. Results show that Topk Pooling is indeed the best among the reported pooling strategies.

Abstraction Layer and KQV Choices. (table 6c). SFC uses an abstraction layer to improve the features of the query $\mathbf{q}$ and the key $\mathbf{k}$ for compression. The features of $\mathbf{v}$ remain instead unchanged, as if they were to be transformed, they would lose compatibility with *tail*. We now evaluate three potential designs: no abstraction layer (1st row, as in self-attention), abstraction layer's transformation for all KQV (2nd row) and our design (3rd row). Results show that transforming $\mathbf{v}$ has a detrimental effect to the model performance, as we conjectured. Furthermore, improving the features of $\mathbf{q}$ and $\mathbf{k}$ for compression using our abstraction layer is very beneficial and it improves Top1 accuracy by 1.9.

Method	Input Size	Usage	FLOPS	Top1
Single Crop	64×1	100%	54G	64.8
Uniform Sampling	64×30	100%	1620G	76.7
Dense Sampling	64×180	100%	9720G	78.2
SFC, $\tau = 2$	256×1	50%	247G	77.4
SFC, $\tau = 4$	256×1	25%	195G	75.2
SFC, $\tau = 8$	256×1	12.5%	167G	68.2
SFC, $\tau = 2$	1024×1	50%	988G	**78.5**
SFC, $\tau = 4$	1024×1	25%	780G	77.2
SFC, $\tau = 8$	1024×1	12.5%	668G	74.0

Table 7: *Results on ActivityNet v1.3 using Slow I3D 8×8.*

Convolution Designs (table 6d). We evaluate different linear transformations for $\mathbf{q}$ and $\mathbf{k}$ in SFC. Results show that employing 3D temporal kernels only in the temporal channel is the best option, which is coherent with the goal of SFC of temporal compression.

6. SFC on Untrimmed Videos

Although SFC is proposed for trimmed videos, we now explore it in the context of untrimmed content. The main challenge with these videos is that their temporal length varies drastically, from a few seconds to potentially many hours. Our SFC model can however take as input approximately 4k frames (∼2 minutes of videos at 30FPS) on modern hardware. To overcome this limitation, we combine SFC with a uniform sampling strategy.

We experiment on ActivityNet v1.3 [3] using a Slow-Only I3D 8×8 backbone. Since this dataset contains videos that are on average shorter than 10 minutes, we uniformly sample 8 clips from each video. Then, we sample a set of consecutive frames from each clip, concatenate these together and feed them into our network. To understand how the number of sampled frames effects the performance, we experiment with two input sizes: 256 and 1024 frames. With 256, we sample 32 frames from each of the 8 video clips, while with 1024 we sample 128 frames (table 7). For reference, we also report results using Single, Uniform and Dense sampling. We also experiment with different compression rates (τ) to evaluate how the performance drops as SFC compresses more and more aggressively.

Results show that SFC improves over Uniform sampling performance using only 15% of its FLOPS (76.7 US vs 77.4 SFC with $\tau = 2$ and input of 256). SFC can also outperform Dense sampling while improving its FLOP by an order of magnitude (78.2 DS vs 78.5 SFC with $\tau = 2$ and input of 1024). While SFC was designed to improve short clip classification performance, these results show that it can also be extended to longer videos. Importantly, note how SFC can be easily adapted to even longer videos than those in ActivityNet by increasing the number of uniformed sampled clips and reducing the number of frames sampled from each clip,

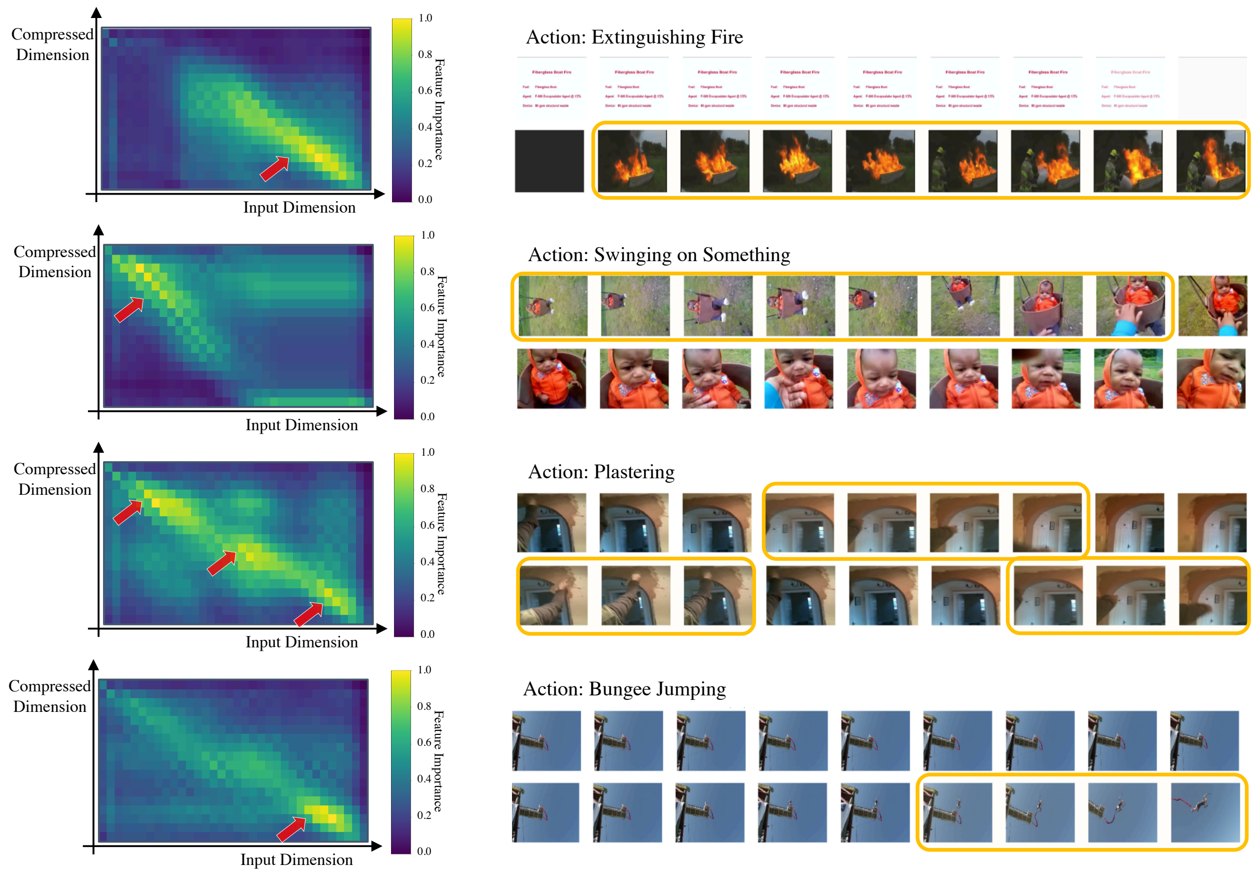

Figure 6: *We show the temporal attention map* $\mathbf{M}$ *(eq. 4) and the corresponding input frames, for four examples. On all videos, SFC is able to attend to the frames relevant for action recognition. For example, only the the second half of the first video contains a fire, and only the first half of the second video contains a baby swinging.*

as our results show that using 256 frames as input already achieves very competitive performance compared to sampling 1024 (77.4 vs 78.5).

7. Visualization

We show four examples in figure 6 to better interpret the functionality of SFC module. In each example, we visualize the temporal association map $\mathbf{M}$ together with the raw frames. Yellow pixels correspond to higher feature responses, which indicate important regions. On the right side, we show these informative frames enclosed in yellow boxes. The first two examples contain background noise and/or shot transitions and our SFC module is able to ignore this irrelevant information. In the last two examples the background is not changing much, yet SFC is able to choose the regions with the clearest motion pattern and with the most important semantic information. For example, in the plastering video, the affinity map responds highly in the temporal region where a hand appears. Moreover, in the bungee jumping video, the affinity map responds highly on the temporal region where the actor starts to jump. All these examples show how our SFC module can indeed select important feature successfully.

8. Conclusion

We presented Feature Compression Plugin (SFC), a flexible plugin that can be inserted into several existing action recognition networks to perform video level prediction in one single pass. Our compression design is guided by the idea that dropping non-informative feature can boost inference speed and not hurt model performance. Our experiments on Kinetics-400, UCF101 and ActivityNet showed that SFC is able to reduce inference speed by 6-7x and memory usage by 5-6x compared with the commonly used 30 crop dense sampling procedure, while also slightly improving Top1 Accuracy.

References

[1] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. Less is more: trading a little bandwidth for ultra-low latency in the data center. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI)*, pages 253–266, 2012. 2

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017. 1, 2, 3, 4, 5

[3] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015. 2, 4, 7

[4] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–213, 2020. 1, 2, 3

[5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019. 1, 2, 3, 4

[6] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12046–12055, 2019. 4, 5

[7] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2069–2077, 2014. 2

[8] Yueyu Hu, Chunhui Liu, Yanghao Li, and Jiaying Liu. Temporal perceptive network for skeleton-based action recognition. In *BMVC*, 2017. 2

[9] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Action localization with tubelets from motion. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 740–747, 2014. 2

[10] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 2

[11] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2760–2769, 2018. 2

[12] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6232–6242, 2019. 1, 2, 3, 5, 6

[13] Xinyu Li, Bing Shuai, and Joseph Tighe. Directional temporal modeling for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 275–291. Springer, 2020. 2

[14] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 909–918, 2020. 2

[15] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 7083–7093, 2019. 2, 5

[16] Chunhui Liu, Yanghao Li, Yueyu Hu, and Jiaying Liu. Online action detection and forecast via multitask deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1702–1706, 2017. 2

[17] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 2

[18] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5058–5066, 2017. 1, 2

[19] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 202–211, 2017. 2

[20] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5533–5541, 2017. 1, 2, 3

[21] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012. 2, 4

[22] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 140–153. Springer, 2010. 1, 2, 3

[23] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[24] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018. 1, 2, 3, 4

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 3

[26] Limin Wang, Yu Qiao, Xiaoou Tang, and Luc Van Gool. Actionness estimation using hybrid fully convolutional net-

works. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[27] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2016. 2, 5, 6

[28] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018. 1, 2, 3, 4, 5

[29] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, and Shilei Wen. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6222–6231, 2019. 1, 2, 6

[30] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321. Springer, 2018. 1, 2, 3

[31] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 591–600, 2020. 2, 4

[32] Sheng Yu, Yun Cheng, Li Xie, and Shao-Zi Li. Fully convolutional networks for action recognition. *IET Computer Vision*, 11(8):744–749, 2017. 2

[33] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2718–2726, 2016. 2

[34] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712. Springer, 2018. 2