3D CNNs with Adaptive Temporal Feature Resolutions

Mohsen Fayyaz[1,*], Emad Bahrami[1,*],
Ali Diba[2], Mehdi Noroozi[3], Ehsan Adeli[4], Luc Van Gool[2,5], Juergen Gall[1]
[1]University of Bonn, [2]KU Leuven, [3]Bosch Center for Artificial Intelligence,
[4]Standford University, [5]ETH Zürich
{lastname}@iai.uni-bonn.de, emadbahramirad@gmail.com,
{firstname.lastname}@kuleuven.be, mehdi.noroozi@de.bosch.de, eadeli@stanford.edu

Abstract

While state-of-the-art 3D Convolutional Neural Networks (CNN) achieve very good results on action recognition datasets, they are computationally very expensive and require many GFLOPs. While the GFLOPs of a 3D CNN can be decreased by reducing the temporal feature resolution within the network, there is no setting that is optimal for all input clips. In this work, we therefore introduce a differentiable Similarity Guided Sampling (SGS) module, which can be plugged into any existing 3D CNN architecture. SGS empowers 3D CNNs by learning the similarity of temporal features and grouping similar features together. As a result, the temporal feature resolution is not anymore static but it varies for each input video clip. By integrating SGS as an additional layer within current 3D CNNs, we can convert them into much more efficient 3D CNNs with adaptive temporal feature resolutions (ATFR). Our evaluations show that the proposed module improves the state-of-the-art by reducing the computational cost (GFLOPs) by half while preserving or even improving the accuracy. We evaluate our module by adding it to multiple state-of-the-art 3D CNNs on various datasets such as Kinetics-600, Kinetics-400, mini-Kinetics, Something-Something V2, UCF101, and HMDB51.

1. Introduction

In recent years, there has been a tremendous progress for video processing in the light of new and complex deep learning architectures, which are based on variants of 3D Convolutional Neural Networks (CNNs) [24, 9, 7, 4, 6, 12, 8]. They are trained for a specific number of input frames, typically between 16 to 64 frames. For classifying a longer video, they slide over the video and the outputs are then aggregated. These networks, however, are often very expensive to train and heavy to deploy for inference task. In order to reduce the inference time, [15, 20] proposed to process not all parts of a video with the same 3D CNN. While [15] trains a second network that decides for each chunk of input frames if it should be processed by the more expensive 3D CNN, [20] uses a fix scheme where a subset of the input chunks are processed by an expensive 3D CNN and the other chunks by a less expensive 3D CNN. The latter then uses an RNN to fuse the outputs of the different 3D CNNs. Although both approaches effectively reduce the GFLOPS during inference, they increase the training time since two instead of one network need to be trained. Furthermore, they do not reduce the computational cost of the 3D CNNs themselves.

In this work, we propose an approach that makes 3D CNNs more efficient for training and inference. Our proposal is based on the observation that the computational cost of a 3D CNN depends on the temporal resolution it operates on at each stage of the network. While the temporal resolution can be different at different stages, the schemes that define how the temporal resolution is reduced is hard-coded and thus the same for all videos. However, it is impossible to define a scheme that is optimal for all videos. If the temporal resolution is too much reduced, the network is forced to discard important information for some videos. This results in a decrease of the action recognition accuracy performance. Vice versa, a high temporal resolution results in highly redundant feature maps and increases the computational time, which makes the 3D CNN highly inefficient for most videos. In this work, we therefore address the question of how a 3D CNN can dynamically adapt its computational resources in a way such that not more resources than necessary are used for each input chunk.

In order to address this question, we propose to exploit the redundancy within temporal features such that 3D

*Mohsen Fayyaz and Emad Bahrami equally contributed to this work. Emad Bahrami contributed to this project while he was a visiting researcher at the Computer Vision Group of the University of Bonn.

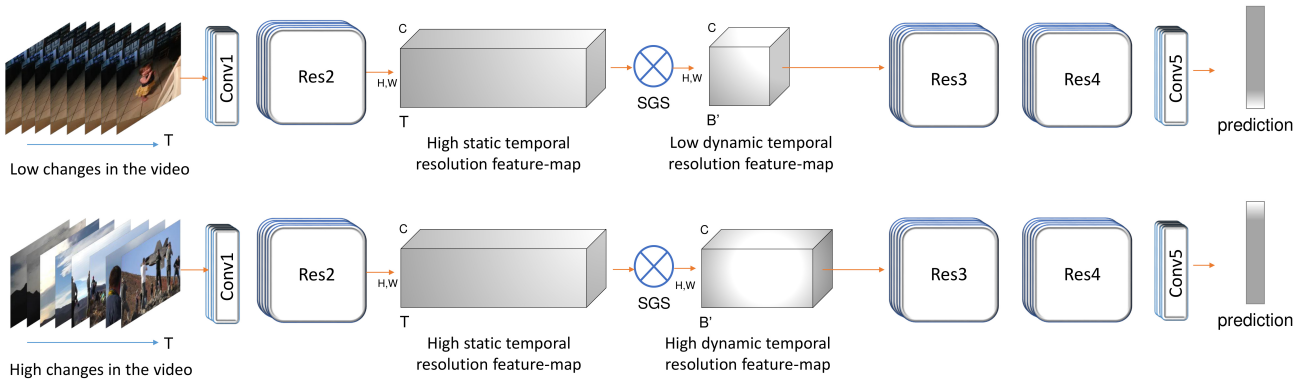

Figure 1: The difficulty of recognizing actions varies largely across videos. For videos with slow motion (top), the temporal features that are processed within a 3D CNN can be highly redundant. However, there are also very challenging videos where all features are required to understand the content (bottom). While previous 3D CNNs use fix down-sampling schemes that are independent of the input video, we propose a similarity guided sampler that groups and aggregates redundant information of temporal features into $B' \leq T$ feature maps. The core aspect is that this process adapts the internal temporal resolution to the input video such that B' is small if the input features are redundant (top) and large (bottom) if most of the features are required.

CNNs process and select the most valuable and informative temporal features for the action classification task. In contrast to previous works, we propose to dynamically adapt the temporal feature resolution within the network to the input frames such that on one hand important information is not discarded and on the other hand no computational resources are wasted for processing redundant information. To this end, we propose a *Similarity Guided Sampling (SGS)* mechanism that measures the similarity of temporal feature maps, groups similar feature maps together, and aggregates the grouped feature maps into a single output feature map. The similarity guided sampling is designed such that it is differentiable and number of output feature maps varies depending on the redundancy of the temporal input feature maps as shown in Fig. 1. By integrating the similarity guided sampling as an additional module within any 3D CNN, we convert the 3D CNN with fixed temporal feature resolutions into a much more efficient dynamic 3D CNN with *adaptive temporal feature resolutions (ATFR)*. Note that this approach is complementary to [15, 20] and the two static 3D CNNs used in these works can be replaced by adaptive 3D CNNs. However, even with just a single 3D CNN with adaptive temporal feature resolutions, we already achieve a higher accuracy and lower GFLOPs performance compared to [15, 20].

We demonstrate the efficiency of 3D CNNs with adaptive temporal feature resolutions by integrating the similarity guided sampler into the current state-of-the-art 3D CNNs such as R(2+1)D [25], I3D [3], and X3D [8]. It drastically decreases the GFLOPs by about half in average while the accuracy remains nearly the same or gain improvements. In summary, the similarity guided sampler is capable of significantly scaling down the computational cost of off-the-shelf 3D CNNs and therefore plays a crucial role for real-world video-based applications.

2. Related Work

The computer vision community has made huge progress in several challenging vision tasks by using CNNs. In recent years, there has been a tremendous progress for video processing in the light of new and complex deep learning architectures, which are based on variants of 3D CNNs [24, 9, 7, 4, 6, 12, 8]. Tran et al. [24] and Carreira et al. [3] proposed 3D versions of VGG and Inception architectures for large-scale action recognition benchmarks like Sports-1M [13] and Kinetics [14]. These methods could achieve superior performance even without using optical-flow or any other pre-extracted motion information. This is due to the capability of 3D kernels to extract temporal relations between sequential frames. Recently, methods like HATNet [5], STC [4], and DynamoNet [6] focus on exploiting spatial-temporal correlations in a more efficient way or on learning more accurate motion representations for videos. These works based on 3D CNNs, however, require huge computational resources since they process sequences of frames with an immense number of 3D convolution layers. There has been therefore a good effort to propose more efficient architectures based on 2D and 3D CNNs [18, 17, 25, 31, 34]. For instance, Lin et al. [18] introduced a temporal shift module (TSM) to enhance 2D-

ResNet CNNs for video classification. The model even runs on edge devices. In [25, 31] 2D and 3D convolutional layers are combined in different ways. SlowFast [9] has explored the resolution trade-offs across temporal, spatial, and channel access. It decreases the computation cost by employing a light pathway with a high temporal resolution for temporal information modeling and a heavy low temporal resolution pathway for spatial information modeling. In relation to this work, [8] investigates whether the light or heavy model is required and presents X3D as a family of efficient video networks.

In order to reduce the inference time of existing networks, [29, 30, 15, 20] proposed to process not all parts of a video with the same CNN model. This line of research is built upon the idea of big-little architecture design. In the context of 2D CNNs, [29, 30] process salient frames by expensive models and use light models to process the other frames. In contrast, [15, 20] do not process single frames but process short chunks of frames with 3D CNNs. [15] trains a second lighter network that decides for each chunk of input frames if it should be processed by the more expensive 3D CNN. [20] uses a fix scheme where a subset of the input chunks are processed by an expensive 3D CNN and the other chunks by a less expensive 3D CNN. It then uses an RNN to fuse the outputs of the different 3D CNNs. Although such approaches effectively reduce the GFLOPS during inference, they increase the training time since two instead of one network need to be trained. Furthermore, they do not reduce the computational cost of the 3D CNNs themselves.

There are various efforts on temporal action detection or finding action segments in untrimmed videos like [1, 23, 33, 28]. These works focus on localizing actions but not on improving the computational efficiency for action recognition. While these works are not related, they can benefit from our approach by integrating the proposed similarity guided sampling module into their CNNs for temporal action localization.

3. Adaptive Temporal Feature Resolutions

Current state-of-the-art 3D CNNs operate at a static temporal resolution at all levels of the network. Due to the redundancy of neighbouring frames, traditional 3D CNN methods often down-sample the temporal resolution inside the network. This helps the model to operate at a lower temporal resolution and hence reduces the computation cost. The down-sampling, however, is static which has disadvantages in two ways. First, a fixed down-sampling rate can discard important information, in particular for videos with very fast motion as it is for instance the case for ice hockey games. Second, a fixed down-sampling rate might still include many redundant temporal features that do not contribute to the classification accuracy as it is for instance the case for a video showing a stretching exercise. We therefore propose a module that dynamically adapts the temporal feature resolution within the network to the input video such that on one hand important information is not discarded and on the other hand no computational resources are wasted for processing redundant information.

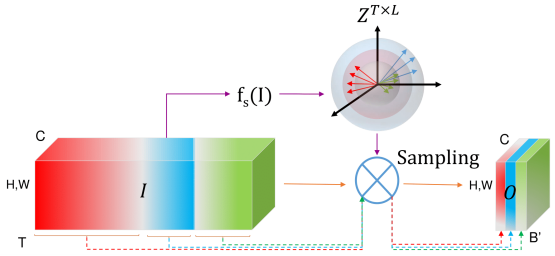

Figure 2: To learn the similarity of the feature maps, we map each temporal feature map $\mathcal{I}_t$ using $f_s(\mathcal{I})$ into an L-dimensional similarity space. After the mapping, $\mathcal{Z} \in \mathbb{R}^{T\times L}$ contains all of the feature maps represented as vectors in the similarity space. Afterwards, we group similar vectors by creating B similarity bins. Using the similarity bins, sampler aggregates the similar feature maps of each bin into the output feature map $\mathcal{O}_b$.

Fig. 1 illustrates a 3D CNN with *adaptive temporal feature resolutions (ATFR)*. The core aspect of ATFR is to fuse redundant information from a temporal sequence of features and extract only the most relevant information in order to reduce the computational cost for processing a video. An important aspect is that this approach is not static, *i.e.*, the amount of information that is extracted varies for each video as illustrated in Fig. 1. In order to achieve this, we propose a novel *Similarity Guided Sampling (SGS)* mechanism that will be described in Sec. 4.

In principle, any 3D CNN can be converted into a CNN with adaptive temporal feature resolutions by using our SGS module. Since the module is designed to control the temporal resolution within the network for each video, it should be added to the early stages of a network in order to get the best reduction of computational cost. For R(2+1)D [25], for example, we recommend to add SGS after the second ResNet block. This means that the temporal resolution is constant for all videos before SGS, but it dynamically changes after SGS. We discuss different 3D CNNs with SGS in Sec. 5.1.

4. Similarity Guided Sampling

The *SGS* is a differentiable module to sample spatially similar feature maps over the temporal dimension and aggregate them into one feature map. Since the number of output feature maps is usually lower than the input feature maps, i.e., $B' < T$, redundant information is removed as il-

lustrated in Figure 2. The important aspect is that B' is not constant, but it varies for each video. In this way, we do not remove any information if there is no redundancy among the input feature maps.

This means that we need to a) learn the similarity of feature maps, b) group similar feature maps, and c) aggregate the grouped feature maps. Furthermore, all these operations need to be differentiable. We denote an input feature map for frame t by $\mathcal{I}_t \in \mathbb{R}^{C\times H\times W}$, where C, H, and W denote the number of channels, height, and width, respectively. To learn the similarity of the feature maps, we map each feature map $\mathcal{I}_t$ into an L-dimensional similarity space. This mapping $f_s(\mathcal{I}_t)$ is described in Sec. 4.1. After the mapping, $\mathcal{Z} \in \mathbb{R}^{T\times L}$ contains all feature maps in the similarity space, which are then grouped and aggregated into B' feature maps. The grouping of $\mathcal{Z}_t$ is described in Sec. 4.2 and the aggregation of the grouped features in Sec. 4.3.

4.1. Similarity Space

The similarity space is a L dimensional vector space where each temporal input feature map is represented by a vector $\mathcal{Z}_t$. The mapping is performed by the similarity network $f_s(\mathcal{I})$ that consists of a global average pooling layer and two convolutional layers. The pooling is applied over the spatial dimension of the feature map while keeping the temporal dimension. Afterward two $1D$ convolutional layers are applied with kernel sizes of 1 and output channel sizes C and L, respectively.

4.2. Similarity Bins

To group similar feature maps $\mathcal{I}_t$, we use the magnitude of each vector $\mathcal{Z}_t$, i.e.,

$$\Delta_t = ||\mathcal{Z}_t|| \tag{1}$$

and we consider two feature maps $\mathcal{I}_t$ and $\mathcal{I}_{t'}$ similar if the value of Δ_t and $\Delta_{t'}$ lie inside a similarity bin. To make the grouping very efficient and differentiable, we propose a binning approach with B similarity bins. We set $B = T$ such that no information is discarded if there is no redundancy between the feature maps of all frames. For most videos, a subset of bins remain empty and will be discarded such that the remaining bins, B', will be less than B as it is described in Sec. 4.3.

We first estimate the half of the width of each similarity bin γ, by computing the maximum magnitude Δ_{max} and dividing it by the number of the desired bins B:

$$\Delta_{max} = \max(\Delta_1, \dots, \Delta_T), \;\; \gamma = \frac{\Delta_{max}}{2B}. \tag{2}$$

Having the width of the similarity bins, the center of each bin β_b is estimated as follows:

$$\beta_b = (2b-1)\gamma \quad \forall b \in (1, \dots, B). \tag{3}$$

4.3. Differentiable Bins Sampling

The grouping and aggregation of all feature maps $\mathcal{I}_t$ based on the bins B will be done jointly by sampling temporal feature maps which belong to the same similarity bin and add them together. We denote the aggregated feature maps for each bin b by $\mathcal{O}_b \in \mathbb{R}^{C\times H\times W}$. To make the process differentiable, we use generic differentiable sampling kernels $\Psi(., \beta_b)$ that are defined such that a sampler only samples from the input temporal feature map $\mathcal{I}_t$ if Δ_t lies in the similarity bin b. This can be written as:

$$\mathcal{O}_b = \sum_{t=1}^{T} \mathcal{I}_t \Psi(\Delta_t, \beta_b). \tag{4}$$

Theoretically, any differentiable sampling kernel that has defined gradients or sub-gradients with respect to Δ_t can be used. In our experiments, we evaluate two sampling kernels. The first kernel is based on the Kronecker-Delta function δ:

$$\mathcal{O}_b = \frac{1}{\sum_{t=1}^{T} \delta\left(\left\lfloor \frac{|\Delta_t - \beta_b|}{\gamma} \right\rfloor\right)} \sum_{t=1}^{T} \mathcal{I}_t \delta\left(\left\lfloor \frac{|\Delta_t - \beta_b|}{\gamma} \right\rfloor\right). \tag{5}$$

The kernel averages the feature maps that end in the same bin. As second kernel, we use a linear sampling kernel:

$$\mathcal{O}_b = \sum_{t=1}^{T} \mathcal{I}_t \max\left(0, 1 - \frac{|\Delta_t - \beta_b|}{\gamma}\right). \tag{6}$$

The kernel gives a higher weight to feature maps that are closer to β_b and less weights to feature maps that are at the boundary of a bin. While we evaluate both kernels, we use the linear kernel by default.

After the sampling, some bins remain empty, i.e., $\mathcal{O}_b = 0$. We drop the empty bins and denote by B' the bins that remain. Note that B' varies for each video as illustrated in Fig. 1. In our experiments we show that the similarity guided sampling can reduce the GFLOPS of a 3D CNN by over 47% in average, making 3D CNNs suitable for applications where they are computationally expensive.

4.4. Backpropagation

Using differentiable kernels for sampling, gradients can be backpropagated through both $\mathcal{O}$ and Δ, where Δ is the magnitude of the similarity vectors $\mathcal{Z}$ which are the outputs of $f_s(.)$. Therefore, we can backpropagate through $f_s(.)$. For the linear kernel (6), which we use if not otherwise specified, the gradient with respect to $\mathcal{I}_t$ is given by

$$\frac{\partial \mathcal{O}_b}{\partial \mathcal{I}_t} = \max\left(0, 1 - \frac{|\Delta_t - \beta_b|}{\gamma}\right) \tag{7}$$

and the gradient with respect to Δ_t is given by

$$\frac{\partial \mathcal{O}_b}{\partial \Delta_t} = \mathcal{I}_t \begin{cases} 0 & |\beta_b - \Delta_t| \geq \gamma \\ \frac{1}{\gamma} & \beta_b - \gamma < \Delta_t \leq \beta_b \\ -\frac{1}{\gamma} & \beta_b < \Delta_t < \beta_b + \gamma \end{cases}. \quad (8)$$

Note that for computing the sub-gradients (8) only the kernel support region for each output bin needs to be considered. The sampling mechanism can therefore be efficiently implemented on GPUs.

5. Experiments

We evaluate our proposed method on the action recognition benchmarks Mini-Kinetics [32], Kinetics-400 [14], Kinetics-600 [2], Something-Something-V2 [11], UCF-101 [22], and HMDB-51 [16]. For these datasets, we use the standard training/testing splits and protocols provided by the datasets. For more details and the UCF-101 and HMDB-51 results please refer to the supplementary material.

5.1. Implementation Details

3D CNNs with ATFR. The similarity guided sampling (SGS) is a differentiable module that can be easily implemented in current deep learning frameworks. We have implemented our SGS module as a new layer in PyTorch which can be easily added to any 3D CNN architecture. To better evaluate the SGS, we have added it to various backbones, such as R(2+1)D [25], I3D [3], X3D [8], and a modified 3DResNet. We place our SGS layer on the second stage of the backbone models. Please refer to the supplementary material for more details. For all of the X3D based models, we follow the training, testing, and measurement setting in [8] unless mentioned otherwise. Additional details and code are available online.[1]

Training. Our models on Mini-Kinetics, Kinetics-400, and Kinetics-600 are trained from scratch using randomly initialized weights without any pre-training. However, we fine-tune on Something-Something-V2, UCF-101, and HMDB-51 with models pre-trained on Kinetics-400. We trained our models using SGD with momentum 0.9 and a weight decay of 0.0001 following the setting in [9]. For Kinetics and Mini-Kinetics, we use a half-period cosine schedule [19] with a linear warm-up strategy [10] to adapt the learning rate over 196 epochs of training. During training, we randomly sample 32 frames from a video with input stride 2. For spatial transformations, we first scale the shorter side of each frame with a random integer from the interval between 256 and 320 [26, 9, 21] then we apply a random cropping with size 224×224 to each frame. Furthermore, each frame is horizontally flipped with probability of 0.5.

[1] https://SimilarityGuidedSampling.github.io

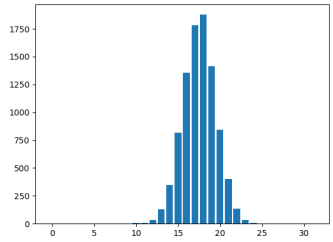

Figure 3: Histogram of active bins for 3DResNet-50 + ATFR on the Mini-Kinetics validation set. The y-axis corresponds to the number of clips and the x-axis to the number of active bins B'.

Testing. We follow [26, 9] and uniformly sample 10 clips from each video for inference. The shorter side of each clip is resized to 256 and we extract 3 random crops of size 256×256 from each clip. For the final prediction, we average the softmax scores of all clips.

Measurements. We report top-1 and top-5 accuracy. To measure the computational efficiency of our models, we report the complexity of the models in GFLOPS based on a single input video sequence of 32 frames and spatial size 224×224 for validation and 256×256 for testing. As shown in Fig. 3, 3D CNNs with ATFR adapt the temporal feature resolutions and the GFLOPs vary for different clips. For ATFR models, we therefore report the average GFLOPs.

5.2. Ablation Experiments

We first analyze different setups for our SGS module. Then, we analyze the efficiency and effect of using our SGS module in different 3D CNN models. If not otherwise specified, we use 3DResNet-18 as 3D CNNs backbones and report the results on the Mini-Kinetics validation set.

5.2.1 Different Similarity Measurements

As mentioned in Sec. 4.2, we use the magnitude of the embedding vectors as the similarity measurement to create the similarity bins. The embedding vectors are represented in an L dimensional space. Instead of magnitudes, we can use other measures such as directions of the vectors. To better study this, we convert the Cartesian coordinates of the vectors to spherical coordinates. In an L dimensional space, a vector is represented by 1 radial coordinate and $L-1$ angular coordinates. To use the spherical coordinates of the vectors for creating the similarity bins, we use multi-dimensional bins and sampling kernels. For more details, please refer to the supplementary material.

We report the results in Table 1. As can be seen, using the magnitudes of the vectors results in a better accuracy compared to angular coordinates or spherical coordinates.

Similarity	Magnitude	Angular	Spherical
top1	**69.6**	68.5	68.7
top5	**88.8**	87.8	88.1

Table 1: Impact of the similarity measure for 3DResNet-18 + ATFR on Mini-Kinetics with linear sampling kernel. We show top-1 and top-5 classification accuracy (%).

Kernel	Linear	Kronecker
top1	**69.6**	68.9
top5	**88.8**	88.6

Table 2: Impact of the sampling kernel.

We believe that due to the similarity of the neighbouring video frames using only magnitudes of the vectors for the similarity measurement is enough and angular or spherical coordinates add too much of complexity to the model. In all of the experiments, the number of bins B is equal to 32. For the angular coordinates, we divide the angles into 4 and 8 bins (4×8). For the spherical coordinates, we divide the radial coordinate into 2 and the angular coordinates into 4 and 4 bins $(2 \times 4 \times 4)$.

5.2.2 Different Sampling Kernels

As mentioned in Sec. 4.3, we can use different differentiable sampling kernels (4). We evaluate two different sampling kernels, namely the Kronecker-Delta sampling kernel (5) and the linear sampling kernel (6). As can be seen in Table 2, the linear kernel performs better than the Kronecker-Delta kernel. The slight superiority of the linear kernel is due to the higher weighting of the temporal feature maps that are closer to the center of the bins. We use the linear kernel for the rest of the paper.

5.2.3 Embedding Dimension

As mentioned in Sec. 4.1, we map the temporal feature maps into an L-dimensional similarity space. In Table 3, we quantify the effect of L. The accuracy increases as L increases until $L = 8$. For $L = 16$ the dimensionality is too large and the similarity space tends to overfit. The model with $L = 1$ is a special case since it can be considered as a direct prediction of Δ_t (1) without mapping the temporal features into a similarity space $\mathcal{Z}_t$. The results show that using a one dimensional embedding space results in a lower accuracy, which demonstrates the benefit of the similarity space.

5.2.4 Different Input Frame-rates

It is an interesting question to ask how a 3D CNN with ATFR performs when the number of input frames or the stride changes for inference. To answer this question, we train two 3D CNNs with ATFR and two without ATFR using 32 input frames and a sampling stride of 2, which corresponds to a temporal receptive field of 64 frames. For inference, we then change the number of frames to 64 and/or the stride to 1.

L	1	4	8	16
top1	67.3	68.4	**69.6**	64.7
top5	87.7	88.1	**88.8**	86.1

Table 3: Impact of the dimensionality L of the similarity space.

model	input frames	GFLOPs	top1		top5	
			stride			
			1	2	1	2
SlowFast-8x8-ResNet18	32	30.9	67.5	69.7	87.1	89.1
	64	61.8 (2.0)	72.1	74.6	89.9	91.9
R(2+1)D	32	46.5	67.4	69.3	86.2	87.5
	64	93.1 (2.0)	70.8	73.7	88.8	91.5
R(2+1)D+ATFR	32	32.3	67.4	69.3	86.4	87.6
	64	54.9 (1.7)	71.4	73.8	88.6	90.7
3DResNet-18+ATFR	32	**14.0**	67.3	69.6	87.2	89.0
	64	**21.1 (1.5)**	72.1	74.8	89.8	91.4

Table 4: Impact of the stride and number of input frames during inference. All models are trained with 32 frames and stride 2.

As it can be seen in Table 4, increasing the input frames from 32 to 64 improves the accuracy of all models. This improvement in accuracy is due to the increase in the temporal receptive field over the input frames while keeping the temporal input resolution. However, the computation cost of the models without ATFR increases as expected by factor 2. If ATFR is used, the increase is only by 1.7 and 1.5 for R(2+1)D+ATFR and 3DResNet-18+ATFR. By comparing R(2+1)D with R(2+1)D+ATFR, we see how ATFR drastically reduces the GFLOPS from 46.5 to 32.3 for 32 frames and from 93.1 to 54.9 for 64 frames. This shows that more frames also increase the redundancy and ATFR efficiently discards this redundancy. Furthermore, it demonstrates that ATFR is robust to changes of the frame-rate and number of input frames.

It is also interesting to compare the results for 32 frames with stride 2 to the results for 64 frames with stride 1. In both cases, the temporal receptive field is 64. We can see the efficiency of our method in adapting the temporal resolution compared to the traditional static frame-rate sampling methods, *i.e.*, 3DResNet-18+ATFR operates on average with 21.1 GFLOPs for 64 input frames compared to SlowFast with GFLOPs of 30.9 (32) and 61.8 (64), and R(2+1)D with GFLOPs of 46.5 (32) and 93.1 (64).

5.2.5 Adaptive Temporal Feature Resolutions

As shown in Fig. 3, the temporal feature resolutions vary for different clips. In order to analyze how the temporal feature resolution relates to the content of a video, we report

Lowest Temporal Resolution	Highest Temporal Resolution
presenting weather forecast	passing American football (in game)
stretching leg	swimming breast stroke
playing didgeridoo	playing ice hockey
playing clarinet	pushing cart
golf putting	gymnastics tumbling

Table 5: The 5 action classes with lowest and highest required adaptive temporal resolution for 3DResNet-50 + ATFR on Mini-Kinetics.

Stage	No SGS	First Conv	Res2	Res3
top1	77.9	77.8	**78.0**	78.0
GFLOPs	1.9	**0.9**	1.1	1.3

Table 6: Evaluating the result of adding our SGS layer to different stages of a X3D-S network on Mini-Kinetics.

in Table 5 the 5 action classes with lowest adaptive temporal feature resolution (<12) and highest adaptive temporal feature resolution (>20). As in Fig. 3, the results are for the 3DResNet-50+ATFR on the Mini-Kinetics validation set. As it can be seen, the actions with less movements like 'presenting weather forecast' result in a low temporal resolution while actions with fast (camera) motions like 'passing American football (in game)' result in a high temporal resolution.

5.2.6 SGS Placement

To evaluate the effect of the location of our SGS module within a 3D CNN, we add it to different stages of X3D-S [8] and train it on Mini-Kinetics. As it can be seen in Table 6, adding SGS to the first stage of X3D-S drastically reduces the GFLOPs by 52.6% (2.1×) while getting slightly lower accuracy. On the other hand, adding SGS after the 2^{nd} stage results in a 42.1% reduction of GFLOPs and slightly higher accuracy. The same accuracy and growth in GFLOPs occurs when SGS is added after the 3^{rd} stage.

5.3. Mini-Kinetics

Mini-Kinetics is a smaller dataset compared to the full Kinetics-400 dataset [14] and consists of 200 categories. Since some videos on YouTube are not accessible, the training and validation set contain 144,132 and 9182 video clips, respectively. Table 7 shows the results on Mini-Kinetics. We add the SGS module to four 3d CNNs R(2+1)D [25], I3D [3], X3D [8], and 3DResNet. In all cases, ATFR drastically reduces the GFLOPS while the accuracy remains nearly the same. For X3D, the accuracy even increases marginally.

5.4. Kinetics-400

We also evaluate ATFR with state-of-the-art 3D CNNs on Kinetics-400 [14], which contains ∼240k training and

model	backbone	GFLOPs	top1	top5
Fast-S3D [32]	-	43.5	78.0	-
SlowFast 8x8	ResNet18	40.4	77.5	93.3
SlowFast 8x8	ResNet50	65.7	79.3	94.2
R(2+1)D	ResNet50	101.8	78.7	93.4
R(2+1)D+**ATFR**	ResNet50	67.3	78.2	92.9
I3D	ResNet50	148.4	79.3	94.4
I3D+**ATFR**	ResNet50	105.2	78.8	93.6
X3D-S	-	1.9	77.9	93.4
X3D-S+**ATFR**	-	**1.1**	78.0	93.5
3DResNet	ResNet50	40.8	79.2	94.6
3DResNet+**ATFR**	ResNet50	**23.4**	79.3	94.6

Table 7: Comparison with state-of-the-art methods on Mini-Kinetics. The accuracy for Fast-S3D [32] is reported with 64 frames.

∼20k validation videos of 400 human action categories. Table 8 shows the comparison with the state-of-the-art. We add the SGS module to the state-of-the-art 3D CNNs SlowFast [9] and three versions of X3D [8].

As it can be seen, our SGS module drastically decreases the GFLOPs of all 3D CNNs. In contrast to Mini-Kinetics, it even improves the accuracy for all 3D CNNs. We will see that this is the case for all large datasets. For X3D-XL [8], we observe a ∼45% reduction in GFLOPs and 0.2% improvement in accuracy. We can see that X3D-XL+ATFR$^{\beta}$ requires similar GFLOPs compared to X3D-L$^{\beta}$ [8] while providing a higher accuracy by 1.8%. We can also see that X3D-XL+ATFR$^{\alpha}$ requires drastically less GFLOPs compared to X3D-L$^{\beta}$[8] while getting a higher accuracy by 1.1%. In comparison to the computational heavy SlowFast16×8,R101+NL [9], X3D-XL+ATFR$^{\beta}$ gets higher top-5 and comparable top-1 accuracy while having 8.9× less GFLOPs.

Comparing the 3D CNNs with ATFR to SCSampler [15] and FASTER [20], which require to train two networks, our approach with a single adaptive 3D CNN achieves a higher accuracy and lower GFLOPs. Note that our approach is complementary to [15, 20] and the two static 3D CNNs used in these works can be replaced by adaptive 3D CNNs. Nevertheless, our approach outperforms these works already with a single 3D CNN. Fig. 4 shows the accuracy/GFLOPs trade-off for a few 3D CNNs with and without ATFR.

5.5. Kinetics-600

We also evaluate our approach on the Kinetics-600 dataset [2]. As shown in Table 9, ATFR shows a similar performance as on Kinetics-400. Our SGS module drastically decreases the GFLOPs of all 3D CNNs while improving their accuracy. For X3D-XL[8], we observe a ∼47.1% reduction of GFLOPs and a slight improvement in accuracy. The best model X3D-XL+ATFR achieves state-of-the-art accuracy. Note that the average GFLOPs of X3D-

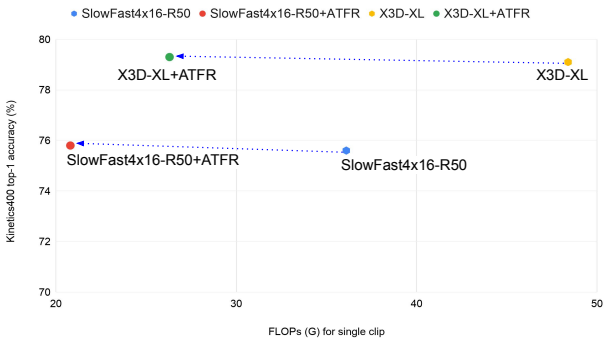

Figure 4: Accuracy vs. GFLOPs for the Kinetics-400 validation set.

model	GFLOPs	top1	top5	Param
I3D*[3]	108× N/A	71.1	90.3	12.0M
I3D+SCSampler*[15]	108×10+N/A	75.1	N/A	N/A
Two-Stream I3D*[3]	216×N/A	75.7	92.0	25.0M
Two-Stream S3D-G*[32]	143×N/A	77.2	-	23.1M
TSM R50*[18]	65×10	74.4	N/A	24.3M
HATNET[5]	N/A	77.2	N/A	N/A
STC[4]	N/A	68.7	88.5	N/A
Two-Stream I3D[3]	216×N/A	75.7	92.0	25.0M
R(2+1)D[25]	152×115	72.0	90.0	63.6M
Two-Stream R(2+1)D[25]	304×115	73.9	90.9	127.2M
FASTER32[20]	67.7×8	75.3	N/A	N/A
SlowFast8×8,R101+NL[9]	116×30	78.7	93.5	59.9M
SlowFast16×8,R101+NL[9]	234×30	**79.8**	93.9	59.9M
X3D-L$^{\alpha}$[8]	18.3×10	76.8	92.5	6.1M
X3D-L$^{\beta}$[8]	24.8×30	77.5	92.9	6.1M
SlowFast4×16,R50[9]	36.1×30	75.6	92.1	34.40M
SlowFast4×16,R50**+ATFR**	20.8 × 30 (↓ 42%)	75.8	92.4	34.40M
X3D-S$^{\alpha}$[8]	1.9×10	72.9	90.5	**3.79M**
X3D-S**+ATFR**$^{\alpha}$	**1.0** × 10 (↓ 47%)	73.5	91.2	**3.79M**
X3D-XL$^{\alpha}$[8]	35.8×10	78.4	93.6	11.09M
X3D-XL**+ATFR**$^{\alpha}$	20 × 10 (↓ 44%)	78.6	93.9	11.09M
X3D-XL$^{\beta}$[8]	48.4×30	79.1	93.9	11.09M
X3D-XL**+ATFR**$^{\beta}$	26.3 × 30 (↓ 45%)	79.3	**94.1**	11.09M

Table 8: Comparison to the state-of-the-art on Kinetics-400. X3D XL+AFTR$^{\beta}$ achieves the STA top5 while requiring 8.8× less GFLOPs compared to STA SlowFast16×8,R101+NL. Following [8], we apply two testing strategies: $^{\alpha}$ samples uniformly 10 clips; $^{\beta}$ takes additionally 3 spatial crops for each sampled clip. For both setups, spatial scaling and cropping settings are as in [8]. * denotes models pretrained on ImageNet.

XL+ATFR are even lower on Kinetics-600 compared to Kinetics-400. This shows that the additional videos of Kinetics-600 are less challenging in terms of motion, which is also reflected by the higher classification accuracy. Compared to SlowFast16×8, R101+NL [9], it requires about 9× less GFLOPs.

5.6. Something-Something-V2

We finally provide results for the Something-Something V2 dataset [11]. It contains 169K training and 25K validation videos of 174 action classes that require more temporal modeling compared to Kinetics. Following [27], we use a R50-SlowFast model pre-trained on Kinetics-400 with 64 frames for the fast pathway, speed ratio of $\alpha = 4$, and channel ratio $\beta = 1/8$. Similar to Kinetics, the SGS module reduces the GFLOPs by 33.9% while keeping the accuracy almost the same. For more implementation details please refer to the supplementary material.

model	pretrain	GFLOPs	top1	top5
Oct-I3D+NL[3]	ImageNet	25.6×30	76.0	N/A
HATNET[5]	HVU	N/A	81.6	N/A
HATNET[5]	-	N/A	80.2	N/A
I3D[3]	-	108× N/A	71.9	90.1
SlowFast16×8,R101+NL[9]	-	234×30	81.8	95.1
SlowFast4×16,R50[9]	-	36.1×30	78.8	94.0
X3D-M[8]	-	6.2×30	78.8	94.5
X3D-M**+ATFR**	-	**3.3** × 30 (↓ 46%)	79.0	94.9
X3D-XL[8]	-	48.4×30	81.9	95.5
X3D-XL**+ATFR**	-	25.6 × 30 (↓ 47%)	**82.1**	**95.6**

Table 9: Comparison to the state-of-the-art on Kinetics-600.

model	pretrain	GFLOPs	top1	top5
SlowFast-R50 [27]	Kinetics400	132.8	61.7	87.8
SlowFast-R50**+ATFR**	Kinetics400	**87.8** (↓ 33%)	**61.8**	**87.9**

Table 10: Results for the Something-Something-V2 dataset.

6. Conclusion

Designing computationally efficient deep 3D convolutional neural networks for understanding videos is a challenging task. In this work, we proposed a novel trainable module called Similarity Guided Sampling (SGS) to increase the efficiency of 3D CNNs for action recognition. The new SGS module selects the most informative and distinctive temporal features within a network such that as much temporal features as needed but not more than necessary are used for each input clip. By integrating SGS as an additional layer within current 3D CNNs, which use static temporal feature resolutions, we can convert them into much more efficient 3D CNNs with *adaptive temporal feature resolutions (ATFR)*. We evaluated our approach on six action recognition datasets and integrated SGS into five different state-of-the-art 3D CNNs. The results demonstrate that SGS drastically decreases the computation cost (GFLOPS) between 33% and 53% without compromising accuracy. For large datasets, the accuracy even increases and the 3D CNNs with ATFR are not only very efficient, but they also achieve state-of-the-art results.

Acknowledgement The work has been financially supported by the ERC Starting Grant ARCA (677650).

References

[1] Humam Alwassel, Fabian Caba Heilbron, and Bernard Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3

[2] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv*, 2018. 5, 7

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 2, 5, 7, 8

[4] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *ECCV*, 2018. 1, 2, 8

[5] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Manohar Paluri, Jürgen Gall, Rainer Stiefelhagen, and Luc Van Gool. Large scale holistic video understanding. In *ECCV*, 2020. 2, 8

[6] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. Dynamonet: Dynamic action and motion network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6192–6201, 2019. 1, 2

[7] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *CVPR*, 2017. 1, 2

[8] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 1, 2, 3, 5, 7, 8

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *ICCV*, 2019. 1, 2, 3, 5, 7, 8

[10] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, 2017. 5

[11] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 5, 8

[12] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *CVPR*, 2018. 1, 2

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[14] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 5, 7

[15] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019. 1, 2, 3, 7, 8

[16] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 5

[17] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 387–403, 2018. 2

[18] 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*, pages 7083–7093, 2019. 2, 8

[19] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[20] Anshul Shah, Shlok Mishra, Ankan Bansal, Jun-Cheng Chen, Rama Chellappa, and Abhinav Shrivastava. Faster recurrent networks for efficient video classification. In *AAAI*, 2020. 1, 2, 3, 7, 8

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 5

[22] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[23] Yu-Chuan Su and Kristen Grauman. Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video. In *European Conference on Computer Vision*, 2016. 3

[24] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2

[25] 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. 2, 3, 5, 7, 8

[26] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 5

[27] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *CVPR*, 2020. 8

[28] 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*, 2019. 3

[29] 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. 3

[30] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *CVPR*, 2019. 3

[31] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018. 2, 3

[32] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 5, 7, 8

[33] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 3

[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, 2018. 2