No frame left behind: Full Video Action Recognition

Xin Liu[1] Silvia L. Pintea[1] Fatemeh Karimi Nejadasl[2] Olaf Booij[2] Jan C. van Gemert[1]
Computer Vision Lab, Delft University of Technology[1] TomTom[2]

Abstract

Not all video frames are equally informative for recognizing an action. It is computationally infeasible to train deep networks on all video frames when actions develop over hundreds of frames. A common heuristic is uniformly sampling a small number of video frames and using these to recognize the action. Instead, here we propose full video action recognition *and consider all video frames. To make this computational tractable, we first cluster all frame activations along the temporal dimension based on their similarity with respect to the classification task, and then temporally aggregate the frames in the clusters into a smaller number of representations. Our method is end-to-end trainable and computationally efficient as it relies on temporally localized clustering in combination with fast Hamming distances in feature space. We evaluate on UCF101, HMDB51, Breakfast, and Something-Something V1 and V2, where we compare favorably to existing heuristic frame sampling methods.*

1. Introduction

Videos have arbitrary length with actions occurring at arbitrary moments. Current video recognition methods use CNNs on coarsely sub-sampled frames [2, 25, 29, 32, 39, 42, 46, 48, 49, 51] because using all frames is computationally infeasible. Sub-sampling, however, can miss crucial frames for action recognition. For example, as shown in Fig. 1, sampling the frame with the dish in the pan is crucial for correct recognition. We propose to do away with sub-sampling heuristics and argue for leveraging all video frames: Full video action recognition.

It is worth analyzing why training CNNs on full videos is computationally infeasible in terms of memory and calculations. The calculations in the forward pass yield activations, while the backward pass calculations give gradients which are summed over all frames to update the weights. Many of these calculations can be done in parallel and thus are well-suited for modern GPUs. When treating videos as a large collection of image frames, the amount of calculations are not too different from those on large image datasets [5].

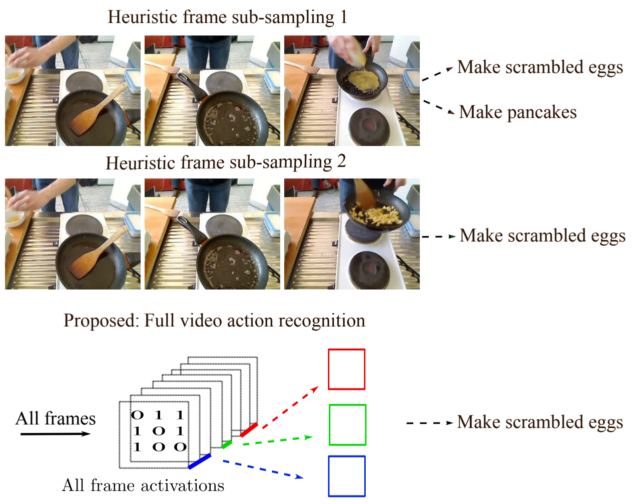

Figure 1. Sub-sampling can miss crucial frames in videos and may cause confusion for action recognition: e.g. compare the two sub-samplings heuristics in row 1 and row 2: Without sampling the dish in the pan it is difficult to classify. Instead, as shown in row 3, we propose to efficiently use all frames during training by clustering frame activations along the temporal dimension and aggregating each cluster to a single representation. The temporal clustering is based on Hamming distances over frame activations, which is computationally fast. With the assumption that similar activations have similar gradients, the aggregated representations approximate the individual frame activations. We efficiently utilize all frames for training without missing important information.

Regarding memory, however, there is a crucial difference between videos and images: The video loss function is not per-frame but on the full video. Hence, to do the backward pass, all activations for each frame, for each filter in each layer need to be stored in memory. This even doubles for storing their gradients. With 10-30 frames per second, this quickly becomes infeasible for even just a few minutes of video. Existing approaches can trade off memory for compute [3, 4, 13] by not storing all intermediate layers, yet they do not scale to video as they would still need to store each frame. The main computational bottleneck for training video CNNs is memory for frame activations.

Here, we propose an efficient method to use all video frames during training. The forward pass computes frame

activations and the backward pass sums the gradients over the frames to update the weights. Now, if only the network was linear, then a huge memory reduction could be gained by first summing all frame activations in the forward pass, which would reduce to just a single update in the backward pass. Yet, deep networks are infamously non-linear, and have non-linearities in the activation function and in the loss function. Thus, if all frames were independent, treating the non-linear network as linear would introduce considerable approximation errors. However, subsequent frames in a video are strongly correlated, and it's this correlation that makes it possible for existing approaches to use sub-sampling. Instead of sub-sampling, we propose to process all frames and exploit the frame correlations to create groups of frames where the network is approximately linear. We use the $ReLU$ (Rectified Linear Unit) activation function, which is linear when the signs of two activations agree, to estimate which parts of the video are approximately linear. This allows us to develop an efficient clustering algorithm based on Hamming distances of frame activations as illustrated in Fig. 1. By then aggregating the approximately linear parts in a video in the forward pass, we make large memory savings in the backward pass while still approximating the full video gradient.
We summarize the contributions of our work as follows:

- We propose a method that allows us to use most or even all video frames for training action recognition by approximated individual frame gradients with the gradients of temporally aggregated frame activations;
- We devise an end-to-end trainable approach for efficient grouping of video frames based on temporally localized clustering and Hamming distances;
- Extensive experiments demonstrate that our method compares well to state-of-the-art methods on several benchmark datasets such as UCF101, HMDB51, Breakfast, and Something-Something V1 and V2.

2. Related work

Action recognition architectures. Actions in video involve motion, leading to deep networks which include optical flow [8, 10, 35], 3D convolutions [2, 6, 15, 21] and recurrent connections [10, 36, 41, 40, 47]. Instead of heavyweight motion representations, a single 2D image can reveal much of an action [20, 23, 35, 42]. 2D CNNs are extremely efficient, and by adding motion information by concatenating a 3D module in ECO [51], modeling temporal relations in TSN [50] or simply shifting filter channels over time in TSM [29] their efficiency is complemented by good accuracy. For this reason, we build on the TSM [29] architecture and modify it for full video action recognition.

Frame sampling for action recognition. Realistic videos contain more frames than can fit in memory. To address this, current methods train by using sub-sampled video frames [2, 29, 42, 51]. Additionally, the SlowFast [7] network also explores the resolution trade-off across temporal, spatial and channel dimension. Rather than using a fixed frame sampling strategy, the sampling can be adaptive [29, 39, 46, 48, 49], or learned to select the best frame [32], or rely on clip sampling [25]. In our work we do not sub-sample frames, but use all frames of the videos, however our clustering is adaptive as it dynamically adapt to the task and the loss function.

Using a subset of frames is computationally more efficient. Using 5-7 frames is sufficient for state-of-the-art action recognition on short videos [33]. Aiming for training efficiency, the work in [43] uses stochastic mini-batch dropping which drops complete batches rather than frames, with a certain probability. Similarly, [45] uses variable mini-batch shapes with different spatio-temporal resolutions varied according to a schedule. Unlike these methods, we do not focus on training efficiency, but propose a method that allows the network to see all video frames during training.

Temporal pooling. To integrate frame-level features, TSN [42] uses average pooling in the late layers of the network. ActionVLAD [11] integrates two-stream networks with a learnable spatio-temporal feature aggregation. Instead of performing temporal pooling or aggregation at a late stage of the network, in [9] RankSVM is used to rank frames temporally and then pool them together. As a follow-up, in [1] a 'dynamic image' is introduced, which is a compact representation of the videos frames using the 'rank pool' operation. In [34, 37] temporal aggregation via pooling and attention is used. Similar to these methods, our proposal performs a temporal pooling of the network activations, however this aggregation is done over clustered activations and it allows us to process all video frames.

Efficient backpropagation. Given that 2/3 of the training computations and memory are spent in the backward pass, existing work focuses on approximations. It is more memory efficient to recompute activations from the previous layer instead of storing them [13], however this comes at the cost of increased training time. In [30] gradient approximations are used where activations are overwritten when new frames are seen without waiting for the backward pass to be performed. Also for efficient backpropagation, randomized automatic differentiation can be used [31], gradients can be reused during training [12], or even quantized during backpropagation [44]. Similar to these works, we use all frames to approximate the full video gradient.

3. Aggregated temporal clusters

3.1. Approximating gradients

We enable the use of all frames of a video during training. To this end, we calculate a single gradient to approximate the gradients of a group of frames. Our hypothesis is that nearby frames in a video are alike, and thus have similar activations, leading to congruent updates. When using the $ReLU$ (Rectified Linear Unit) activation function, we know that for activations with agreeing signs, the activation function is linear. Assuming that similar frames are approximately linear, the standard computation of the sum of gradients over all frames, becomes equivalent to first summing all frame activations and then computing a single gradient. This is computationally and memory efficient. Mathematically, for frames i, this can be formulated as:

$$\sum_i \nabla_{\mathbf{w}} L(h(\mathbf{x}_i \mathbf{w})) = \nabla_{\mathbf{w}} L\left(\sum_i h(\mathbf{x}_i \mathbf{w})\right), \qquad (1)$$

where $\mathbf{x}$ are frame activations, $\mathbf{w}$ are the network weights, $h(\cdot)$ is an activation function, and $L(\cdot)$ is the loss function. Note that Eq. (1) only holds in the ideal case when the activation function h is linear for similar frames and the loss function L is also linear. This is not generally the case, and this approximation introduces an error.

With the above ideal scenario in mind, we can use all video frames without calculating the gradient for each frame, by grouping frames that agree in the sign of their activations $\mathbf{x}$. Over these grouped activations we calculate a single gradient $\nabla_{\mathbf{w}} L(\sum_i h(\mathbf{x}_i \mathbf{w}))$. However, for similar frames the sign of their activation values may not be in complete agreement. Therefore, we aim to find which frames can be safely grouped together, to minimize the error introduced by our approximation in Eq. (1).

3.2. Error bound for the approximation

For ease of explanation, we consider two input video frames and their activations $\mathbf{x}$=$\{\mathbf{x}_1, \mathbf{x}_2\}$, and a convolutional operation with parameters $\mathbf{w}$, denoted by $\mathbf{xw}$. The two frames have the same class label, y, since they come from the same video. We consider a multi-class setting using the cross-entropy loss in combination with the softmax function q, which for these two samples is:

$$L(\mathbf{x}, y) = -\frac{1}{2}\left(\log q_y(\mathbf{x}_1) + \log q_y(\mathbf{x}_2)\right), \qquad (2)$$

where $q_c(\mathbf{x}_i)$=$\frac{\exp(h(\mathbf{x_i w}_c))}{\sum_{j=1}^{C} \exp(h(\mathbf{x_i w}_j))}$, c∈$\{1, .., C\}$ indexes video classes and $h(\cdot)$ is the $ReLU$ activation function. The gradient of the loss with respect to $\mathbf{w}$ is:

$$\nabla_{\mathbf{w}} L(\mathbf{x}, y) = \frac{\mathbf{x}_1(q_c(\mathbf{x}_1) - \delta_{yc}) + \mathbf{x}_2(q_c(\mathbf{x}_2) - \delta_{yc})}{2}, \qquad (3)$$

where δ_{yc} is the Dirac function which is 1 when c=y. In our method, we first average the two activations after the convolution and before the $ReLU$. We can do this, because if we assume the activations have agreeing signs sign($\mathbf{x}_1\mathbf{w}$)=sign($\mathbf{x}_2\mathbf{w}$), then it holds that: $\frac{h(\mathbf{x}_1\mathbf{w})+h(\mathbf{x}_2\mathbf{w})}{2} = h(\frac{\mathbf{x}_1\mathbf{w}+\mathbf{x}_2\mathbf{w}}{2})$. In this case the cross-entropy loss becomes:

$$\hat{L}(\mathbf{x}, y) = -\log q_y\left(\frac{\mathbf{x}_1 + \mathbf{x}_2}{2}\right). \qquad (4)$$

In the backward pass, we calculate a single gradient of the averaged activations as follows:

$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y) = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2}\left(q_c\left(\frac{\mathbf{x}_1 + \mathbf{x}_2}{2}\right) - \delta_{yc}\right), \qquad (5)$$

We now estimate the relative error introduced by our approximation by comparing equations Eq. (3) and Eq. (5) using Jensen's inequality. We start from the softmax function $q_c(\cdot)$ and we recover back equations Eq. (3) and Eq. (5). The softmax function $q_c(\cdot)$ is convex, therefore we can apply to it Jensen's inequality for the samples $\mathbf{x}_1$ and $\mathbf{x}_2$: $q_c\left(\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2}\right) \leq \frac{q_c(\mathbf{x}_1)+q_c(\mathbf{x}_2)}{2}$. We start by considering the case $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2} > 0$. If we multiply both sides of this inequality with $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2}$ we obtain that:

$$\frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} q_c\left(\frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2}\right) \leq \frac{\mathbf{x}_1 q_c(\mathbf{x}_1) + \mathbf{x}_2 q_c(\mathbf{x}_2)}{2} - \frac{1}{4}(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2)). \qquad (6)$$

In the left hand side of the inequality we recover precisely the $\nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y)$ given by Eq. (5), while in the right hand side we recover Eq. (3) minus the approximation error as $\nabla_{\mathbf{w}} L(\mathbf{x}, y) - \frac{1}{4}(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2))$. Note that for the case y=c the additional Dirac terms in $\mathbf{x}$ cancel out. We now consider also the case $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2} \leq 0$, which together with the previous case leads to the following bound on the absolute difference between the gradients in Eq. (3) and Eq. (5):

$$|\nabla_{\mathbf{w}} L(\mathbf{x}, y) - \nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y)| \leq \frac{1}{4}|(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2))|. \qquad (7)$$

Thus, the difference between the two gradient updates is bounded by a function depending on the difference between the activations and their softmax responses. The closer to 0 the difference between the activations the smaller the difference between their gradient updates. We show in the experimental section that, indeed, small differences in the activations entail small differences in the loss.

The inequality in Eq. (6) holds under the condition that the sign of activations agree. Therefore, we want to group frames based on the sign similarity of their activations.

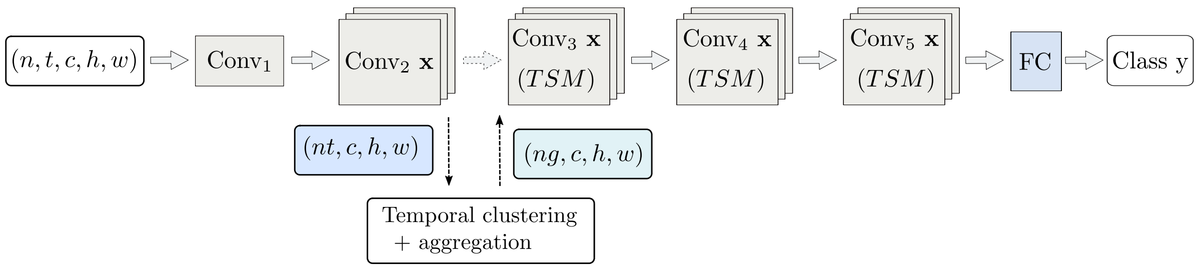

Figure 2. We adopt 2D ResNet-50 with TSM [29] a backbone. The input batch size is n with t frames. We cluster the activations of the first block of size (nt, c, h, w) which groups t frames into g clusters and outputs new activations of size (ng, c, h, w), as input to the next network blocks. Our full video method efficiently utilizes all frames and is end-to-end trainable.

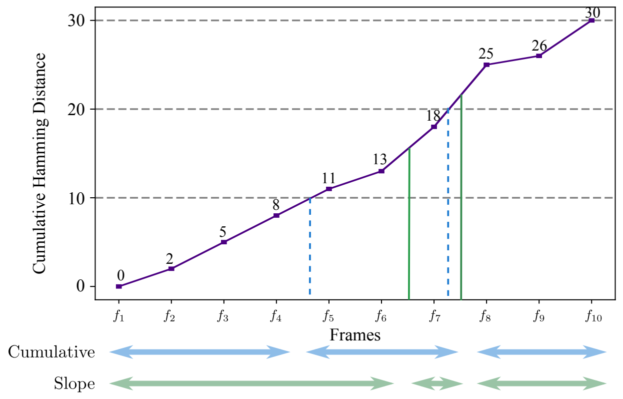

Figure 3. An illustration of our two clustering algorithms. The numbers on the solid line are pair-wise Hamming distances and the solid line is the cumulative Hamming distance of frame f_1 to f_{10}. For g=3 clusters, the *cumulative clustering* groups frames by dividing the total cumulative distance on the y-axis into 3 equally distanced segments, as shown with the dashed lines resulting in the 3 clusters $(f_1 - f_4)$, $(f_5 - f_7)$ and $(f_8 - f_{10})$. The *slope clustering* algorithm is based on the slope of the curve and here selects the top-2 largest slopes, as shown with the solid green lines, which results in the 3 clusters: $(f_1 - f_6)$, (f_7), $(f_8 - f_{10})$.

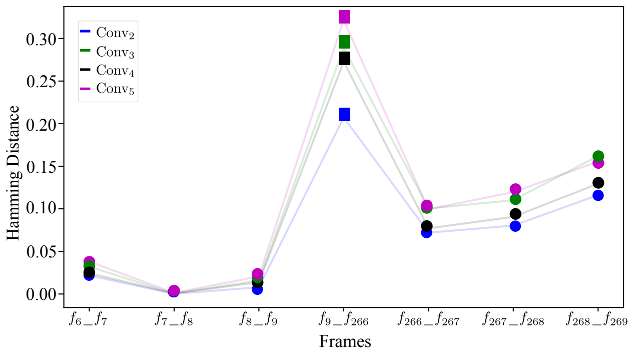

Figure 4. Hamming distances between similar frames and dissimilar frames across 4 blocks of ResNet. The frames are taken from a single Breakfast [26] video. We denote the frames that are similar to their neighbors with circles and the dissimilar ones with squares. Hamming distances are consistent across blocks.

3.3. Temporal clustering and aggregation

Using our proposal in Eq. (5) allows training on all video frames. We group frames based on the sign agreement of their activations. An efficient way to do this, is to binarize the activation values by using the *sign* function and compute a fast Hamming distance between binarized activations to determine which frames to group.

Consecutive frames in a video are more likely to be similar in appearance and are thus more likely to have similarly signed activations. Therefore, we explore two variants of a temporal clustering algorithm based on Hamming distances, where we allow a fixed number of clusters g to match the available memory. We employ the temporal order of video frames and calculate Hamming distances only with neighboring frames. Fig. 3 illustrates the two temporal clustering algorithms we consider here: *cumulative clustering* and *slope clustering*. We start by calculating the cumulative Hamming distance $\mathcal{C}(\mathbf{x})$ for neighboring frames along the temporal order:

$$\mathcal{C}_N(\mathbf{x}) = \sum_{i=1}^{N-1} H(\mathbf{x}_i, \mathbf{x}_{i+1}), \tag{8}$$

where $\mathbf{x}_i$ is the binarized activation of frame i, $H(\cdot,\cdot)$ is the Hamming distance, and N is the total number of frames. For *cumulative clustering*, we divide the total cumulative Hamming distance, $\mathcal{C}(\mathbf{x})$, into g even segments, where the cluster id for frame i is $\lceil g \frac{\mathcal{C}_i(\mathbf{x})}{\mathcal{C}_N(\mathbf{x})} \rceil$. For the *slope clustering*, the boundaries of the segments are defined by the frame indexes corresponding to the top-g largest slopes where the cumulative distance increases the most.

For efficiency, we cluster early on in the network, and input to the subsequent layers only aggregated activations. We assume that the sign of the activations corresponding to two similar frames, approximately agree throughout the network. To validate this, we visualize in Fig. 4 the Hamming distance over activations corresponding to similar and dissimilar frames. The distances corresponding to similar frames remain consistent across different layers.

Putting everything together, we input a set of n videos into our TSM-based [29] network architecture. After the first block, we apply temporal clustering and average the activations within each cluster, giving rise to g activations per video. These aggregated activations are input to the subsequent blocks of the network. Our method efficiently utilizes all frames for training and it is end-to-end trainable, as the gradients propagate directly through the aggregated feature-maps. Fig. 2 depicts the outline of our method.

4. Experiments

We evaluate our method on the action recognition datasets Something-Something V1 & V2 [14], UCF-101 [38], HMDB51 [27] and Breakfast [26]. The consistent improvements show the effectiveness and generality of our method. We validate and analyze our method on a fully controlled Move4MNIST dataset we created. We also include ablation studies of the components of our method.

Datasets. Something-Something V1 [14] consists of 86k training videos and 11k validation videos belonging to 174 action categories. The second release V2 of Something-Something increase the number of videos to 220k. The UCF101 [38] dataset contains 101 action classes and 13,320 video clips. The HMDB51 [27] dataset is a varied collection of movies and web videos with 6,766 video clips from 51 action categories. Breakfast [26] has long videos of human cooking activities with 10 categories with 1,712 videos in total, with 1,357 for training and 335 for testing. Our fully controlled dataset Move4MNIST has four action classes $\{$*move up, move down, move left, move right*$\}$, and 1,800 videos for training and 600 videos for testing. Each video has 32 frames, with a digit from MNIST [28] moving on a UCF-101 video background. To obtain a per-frame ground truth of which frames are relevant we randomly inserted a consecutive chunk of UCF-101 background frames, black frames and frames with MNIST digits that are not part of the target classes. An example is shown in Fig. 7.

Training & Inference. Following the setting in TSM [29], our models are fine-tuned from Kinetics [24] pre-trained weights and we freeze the Batch Normalization [19] layers for HMDB51 [27] and UCF101 [38] datasets. For other datasets, our models are fine-tuned from ImageNet [5] pre-trained weights. To optimize the GPU we train with a fixed number of frames per batch. If the video has less frames, we pad it repeatedly with the last video frame. We compare and cluster the activations of all the frames in each video, and get g average activations for each video, from the first block of our model. We set the number of clusters to $g = \{8, 16\}$ to align with the sub-sampling methods using 8 or 16 frames. During testing, we follow the setting of TSM and sample one clip per video and use the full resolution image with the shorter side 256.

Backbone architecture. For a fair comparison with the state-of-the-art, we evaluate our method on the TSM [29] backbone replying on the ResNet-50 [16] architecture. We use TSM with a ResNet-18 as the backbone for the experiments on our toy dataset Move4MNIST and for model analysis on the Breakfast dataset.

4.1. Are more frames better?

To make it computationally possible to use all individual frames we evaluate on the fully controlled Move4MNIST to test if using more frames during training is better than sub-sampling. We use here the ResNet-18 [16] backbone pretrained on ImageNet [5] and compare with TSM [29]. We evalute slope clustering and cumulative clustering, and a cluster-free uniform grouping of evenly distributed segments and then aggregating them (*Ours-uniform*).

Model	#Frames	#Clusters	FLOPs /Video	Runtime Mem./Video	Top-1
TSM	8	-	14.56G	1.04GB	90.13 ± 0.38
TSM	16	-	29.12G	1.72GB	93.78 ± 0.33
TSM	all	-	58.24G	3.15GB	**98.83 ± 0.16**
Ours-uniform	all	8	28.61G	1.56GB	90.25 ± 0.28
Ours-slope	all	8	28.61G	1.56GB	93.33 ± 0.19
Ours-cumulative	all	8	28.61G	1.56GB	**94.08 ± 0.25**
Ours-uniform	all	16	38.51G	1.79GB	92.73 ± 0.25
Ours-slope	all	16	38.51G	1.79GB	94.06 ± 0.18
Ours-cumulative	all	16	38.51G	1.79GB	**95.27 ± 0.21**

Table 1. Training with all frames gives best accuracy. Our method with slope or cumulative clustering outperforms the uniform grouping of evenly distributed segments and frame subsampling. Our method has less FLOPs and runtime memory usage than TSM training with all frames.

Table 1 shows that TSM trained on all the 32 frames of a video in Move4MNIST significantly outperforms TSM trained on 8 and 16 sub-sampled frames. The uniform grouping of evenly distributed segments does not much better than random sub-sampling, and uniform grouping performs worse than random sub-sampling when the frame and cluster numbers increased from 8 to 16. This can be explained since the videos in the Move4MNIST contain black frames, UCF-101 background frames, and frames containing another digits at random positions, which can make subsampling miss frames related to the task and evenly distributed segments group frames wrongly. Both our clustering approaches with 8 and 16 clusters do better than evenly distributed segments or sub-sampling with 8 or 16 frames as they can adapt to the content and dynamically choose which frames to group. In addition, our method has significantly reduced FLOPs and runtime memory when compared to the baseline on all frames.

4.2. Do similar frames have similar gradients?

In this experiment, we evaluate our assumption that similar frame activations have similar gradients. The activa-

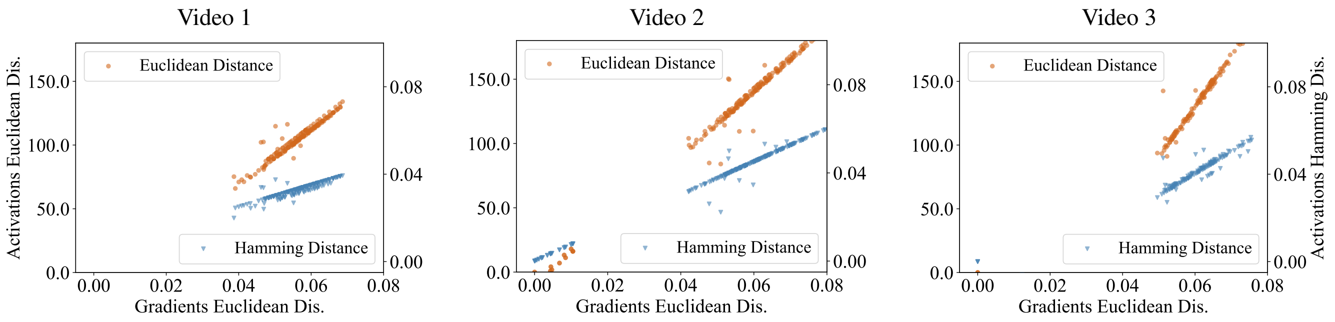

Figure 5. An illustration of activation distance versus gradient distance for frames from three videos in Move4MNIST dataset. For frames that are similar with respect to recognizing the action, the activation distance and the gradients distance between them have a nearly linear relation for both the Euclidean distance and the Hamming distance. Our assumption that frames having similar activations with respect to the task have similar gradients is validated.

tions and gradients are taken from the 1st block of ResNet-18. We show the Euclidean and the Hamming activation distance versus the gradient Euclidean distance between all $32 * 31/2 = 496$ frame pairs for three videos in Move4MNIST in Fig. 5. For both the Euclidean distance and the Hamming distance the relation between activations and gradients is close to linear. It validates our assumption that frames having similar activations with respect to the task have similar gradients.

We compare the ground truth gradients when training truly on all frames to our efficient approximation. We use 16 clusters and compare our approximate gradients to the real gradients which are from 3rd block of ResNet-18 for a video in Move4MNIST. We compare the results of our method with cumulative clustering, slope clustering and uniform grouping. Results in Fig. 6 show that compared to uniform grouping, cumulative clustering and slope clustering give smaller Euclidean distance between the single gradient from each cluster and the sum of gradients of frames in the corresponding cluster. And cumulative clustering gives even smaller gradients differences than slope clustering. In other words, it means that our method with cumulative clustering (the right hand side of Eq. (1)) approximates the standard gradients calculation (the left hand side of Eq. (1)) in the network with a small difference.

4.3. Analyzing model properties

We evaluate the clustering methods, the number of clusters, and the training time efficiency on Breakfast and Move4MNIST with a ResNet-18 backbone.

Different temporal clustering methods. We compare slope clustering, cumulative clustering, and uniform grouping where the videos are split into equal segments. From Table 2, cumulative clustering outperforms slope clustering, while uniform grouping has the lowest top-1 accuracy. This is because equal temporal grouping merges non similar frames together leading to linear approximations of non-linear information and incorrect network updates, resulting in a low action recognition accuracy. A similar trend is also visible on the Move4MNIST dataset in Table 1.

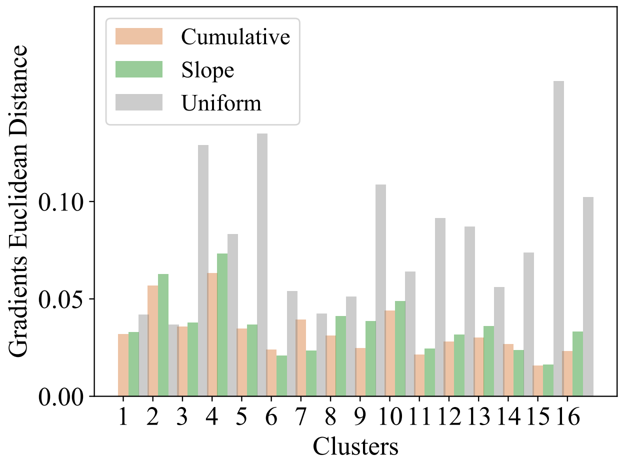

Figure 6. Comparing the Euclidean distance between gradients of the ground truth of truly using all frames to our efficient approximation per cluster for cumulative clustering, slope clustering and uniform grouping on Move4MNIST. Compared to uniform grouping and slope clustering, cumulative clustering results in smaller gradients difference and thus a better approximation.

Model	#Frames	#Clusters	Tr. sec/epoch	Top-1
TSM	8	-	97.6	59.1
TSM	16	-	113.7	61.4
Ours-uniform	all	8	100.1	58.3
Ours-slope	all	8	99.6	60.7
Ours-cumulative	all	8	101.3	62.0
Ours-uniform	all	16	114.0	60.2
Ours-slope	all	16	114.5	63.7
Ours-cumulative	all	16	115.2	**64.4**

Table 2. With 8 and 16 clusters we consistently outperform TSM with 8 and 16 frames for comparable training time on the Breakfast dataset.

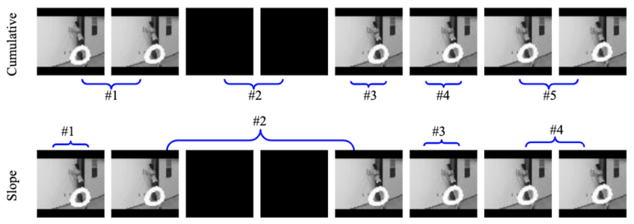

Figure 7. Temporal clustering results for a video in Move4MNIST. Cumulative temporal clustering groups frames more accurately than slope temporal clustering.

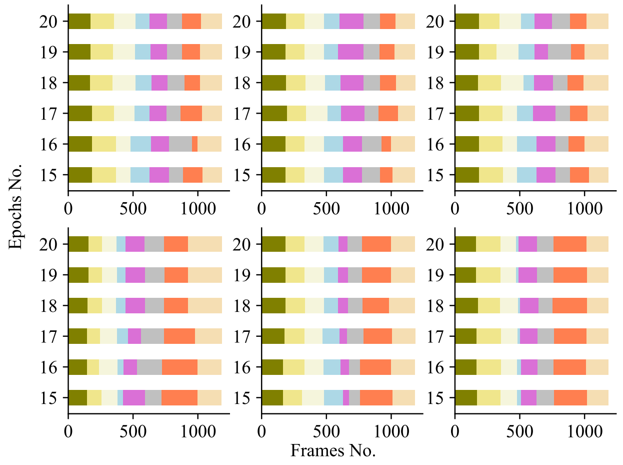

Figure 8. Cumulative temporal clustering results over epochs for six videos in the Breakfast dataset. Each cluster is shown in a different color. Clusters contains segments with different lengths. Our cumulative temporal clustering groups frames with similar activations together. The cluster lengths change according to the changes in the frame activations during training.

In Fig. 7, we show the temporal clustering results for a small number of frames of a Move4MNIST video. Cumulative clustering correctly groups similar frames together, while slope clustering groups moving zero frames and black frames together.

Number of clusters. We conduct experiments using 8 and 16 clusters for our method, which follows the protocol of TSM with 8 and 16 frames for training. Table 2 shows that using 16 clusters consistently outperforms using 8 clusters for all clustering methods. A larger number of clusters improves accuracy. In the extreme case, the cluster numbers equal the number of frames in a video, which is equivalent with using all frames for training. From the table we can also see that our cumulative temporal clustering implementation improves the top-1 accuracy by 2.9% and 3.0%, separately for 8 clusters and 16 clusters comparing to TSM with 8 and 16 frames.

To show that our cumulative temporal clustering algorithm is different from the naive uniform grouping, we visualize the 8 clusters obtained from cumulative temporal clustering for six videos over different epochs in the Breakfast dataset in Fig. 8. Different videos have different segment lengths in the cumulative temporal clustering, which takes the similarity of frame activations into consideration. In Fig. 8, we also show that the cluster length changes over epochs during training, since the activations change during training.

Model	**Backbone**	**#Frames**	**#Clusters**	**Top-1**
ResNet-152[18]	ResNet152	64	-	41.1%
ActionVLAD [18]	ResNet152	64	-	55.5%
VideoGraph [18]	ResNet152	64	-	59.1%
TSM [29] (our impl.)	ResNet50	16	-	72.1%
Ours-slope	ResNet50	all	16	74.9%
Ours-cumulative	ResNet50	all	16	**76.6%**

Table 3. Our method using either slope temporal clustering or cumulative temporal clustering compared to existing works on the Breakfast dataset. Our proposal outperforms TSM, and significantly exceeds in top-1 accuracy methods using the deeper backbone architecture, ResNet-152. By using all frames our method has an advantage on long-term video action recognition.

Efficiency of training time. Table 2 gives the training time per epoch for all the models. Our method with 8 clusters and 16 clusters only has an increase of 3.7 seconds and 1.5 seconds in training time per epoch, when compared to TSM with 8 frames and 16 frames. The results show that our method is efficient during training time, while using all video frames.

4.4. Comparison with the state-of-the-art

We compare our method with the state-of-the-art on Something-Something V1&V2, Breakfast, UCF-101 and HMDB51. All methods use ResNet-50 pre-trained on ImageNet as a backbone, unless specified otherwise.

Comparison on the Breakfast dataset. We compare our method with existing work on the Breakfast dataset, which contains long action videos. Our method using either slope temporal clustering or cumulative temporal clustering largely outperforms the three methods using ResNet-152 as a backbones, in Table 3. Compared to TSM using 16 sub-sampled frames, our method improves the top-1 accuracy by 2.8% and 4.5% with slope temporal clustering and cumulative temporal clustering, respectively. Methods using sub-sampling can easily miss important frames for the recognition task on long action videos. Our method has an advantage on the long videos for action recognition, by efficiently utilizing all the frames.

Comparison on the Something-Something dataset. In Table 5, we list the results of our method compared to other methods on the Something-Something V1&V2 datasets. We achieve state-of-the-art performance on both V1 and

Model	Backbone	Pre-train	#Frames	#Clusters	Top-1 UCF-101	Top-1 HMDB51
TSM [29] (our impl.)	ResNet50	Kinetics	1	-	91.2%	65.1%
TSN [29]	ResNet50	Kinetics	8	-	91.7%	64.7%
SI+DI+OF+DOF [1]	ResNeXt50	Imagenet	dynamic images	-	95.0%	71.5%
TSM [29]	ResNet50	Kinetics	8	-	95.9%	73.5%
STM [22]	ResNet50	ImageNet+Kinetics	16	-	96.2%	72.2%
Ours-slope	TSM-ResNet50	Kinetics	all	8	96.2%	73.3%
Ours-cumulative	TSM-ResNet50	Kinetics	all	8	**96.4%**	**73.4%**

Table 4. Top-1 accuracy on UCF-101 and HMDB51. Our method performs only slightly better than the state-of-the-art on the scene-related datasets UCF-101 and HMDB51. These datasets do not have much frame diversity per video, thus, the improvement of our method over sampling methods is limited.

Model	#Frames	#Clusters	Top-1 V1	Top-1 V2
TSN [29]	8	-	19.7%	30.0%
TRN-Multiscale [29]	8	-	38.9%	48.8%
TSM [29]	8	-	45.6%	59.1%
TSM [22]	16	-	47.2%	63.4%
STM [22]	8	-	49.2%	62.3%
STM [22]	16	-	50.7%	64.2%
Ours-slope	all	8	46.7%	60.2%
Ours-cumulative	all	8	49.5%	62.7%
Ours-cumulative	all	16	**51.4%**	**65.1%**

Table 5. Top-1 accuracy on Something-Something V1 and V2 datasets. Our method using cumulative temporal clustering outperforms the state-of-the-art methods on both Something-Something V1 and V2. Our method achieves limited accuracy improvement for shorter videos.

V2, with outperforming STM of 8 frames by 0.3% and 0.4% for V1 and V2, and STM of 16 frames by 0.7% and 0.9% for V1 and V2 respectively. Comparing to TSM, we significantly improve the top-1 accuracy of 8 frames by 3.9% and 3.5%, and the top-1 accuracy of 16 frames by 4.2% and 1.7% for the V1 and V2 datasets. Although the Something-Something dataset is characterized by temporal variations, the video clips are short compared to the Breakfast dataset. The methods using frame sampling heuristics can capture the main movement in videos. Therefore, our accuracy improvement is not as pronounced as for the Breakfast dataset.

Comparison on the UCF-101 and HMDB51 datasets. We train with 8 clusters and evaluate over three splits and report averaged results in Table 4. Our performance is on par with state-of-the-art methods on both datasets. The UCF-101 and HMDB51 have a scene-bias, where motion plays a limited role and just a few number of frames –or even a single frame– is sufficient. Thus, methods relying on sampling heuristics can correctly classify the actions and our method using all frames is not expected to improve results. To test this, we show results with a single frame in Table 4 which shows that TSM with 1 frame achieves comparable accuracy to TSN with 8 frames on UCF-101 and outperforms TSN with 8 frames on HMDB51. For scene-biased datasets, using all frames does not bring accuracy benefits.

5. Conclusion

We propose an efficient method for training action recognition deep networks without relying on sampling heuristics. Our work offers a solution to using all video frames during training based on the assumption that similar frames have similar gradients, leading to similar parameter updates. To this end, we efficiently find frames that are similar with respect to the classification task, by using a cumulative temporal clustering algorithm based on Hamming distances. The clustering based on Hamming distances enforces that activations in a cluster agree in signs, which is a requirement entailed by our assumption that we can approximate the gradients of multiple frames with a single gradient of an aggregated frame. We accumulate the activations within each cluster to create new representations used to classify the actions. Our proposed method shows competitive results on large datasets when compared to existing work.

Despite our state of the art results, we identify several limitations. One limitation is that the number of clusters is fixed and thus not well-suited for inhomogeneous videos with more semantic (shot) changes than clusters. This could create a dependency for action proposals or other approaches to pre-segment a video in homogeneous segments which somewhat counters the philosophy of using full video action recognition. Another limitation is that for grouping frames the only non-linearity we consider is the activation function and do not use the non-linearity in the loss. This limitation seems insurmountable, as memory constraints prevent us to store all frame activations for when the loss is computed. Nevertheless, with our current results and analysis, we make a first move for action recognition to go full video.

Acknowledgments This work is part of the research program Efficient Deep Learning (EDL), which is (partly) financed by the Dutch Research Council (NWO).

References

[1] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. Action recognition with dynamic image networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2799–2813, 2018. 2, 8, 12

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 2, 12

[3] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI*, 2018. 1

[4] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 1, 5

[6] Tran Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: Generic features for video analysis. *Corr*, 2(8), 2014. 2

[7] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019. 2

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016. 2

[9] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):773–787, 2016. 2

[10] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Two stream lstm: A deep fusion framework for human action recognition. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 177–186. IEEE, 2017. 2

[11] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 971–980, 2017. 2

[12] Negar Goli and Tor M Aamodt. Resprop: Reuse sparsified backpropagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1548–1558, 2020. 2

[13] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pages 2214–2224, 2017. 1, 2

[14] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017. 5

[15] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[17] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 254–263, 2019. 11

[18] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Videograph: Recognizing minutes-long human activities in videos. *ICCV 2019, Workshop on Scene Graph Representation and Learning*, 2019. 7, 11

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 5

[20] Mihir Jain, Jan C Van Gemert, and Cees GM Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 46–55, 2015. 2

[21] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012. 2

[22] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2000–2009, 2019. 8, 11, 12

[23] 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

[24] 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. *CoRR*, 2017. 5

[25] 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*, pages 6232–6242, 2019. 1, 2

[26] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*, 2014. 4, 5

[27] Hilde Kuehne, Hueihan Jhuang, E. Garrote, T. Poggio, and Thomas Serre. Hmdb: A large video database for human motion recognition. *2011 International Conference on Computer Vision*, pages 2556–2563, 2011. 5

[28] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010. 5

[29] 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. 1, 2, 4, 5, 7, 8, 11, 12

[30] Mateusz Malinowski, Grzegorz Swirszcz, Joao Carreira, and Viorica Patraucean. Sideways: Depth-parallel training of video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11834–11843, 2020. 2

[31] Deniz Oktay, Nick McGreivy, Joshua Aduol, Alex Beatson, and Ryan P Adams. Randomized automatic differentiation. *CoRR*, 2020. 2

[32] Jian Ren, Xiaohui Shen, Zhe Lin, and Radomir Mech. Best frame selection in a short video. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3212–3221, 2020. 1, 2

[33] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does human action recognition require? In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 2

[34] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long-range video understanding. In *European Conference on Computer Vision*, pages 154–171, 2020. 2

[35] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 2

[36] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1961–1970, 2016. 2

[37] Sibo Song, Ngai-Man Cheung, Vijay Chandrasekhar, and Bappaditya Mandal. Deep adaptive temporal pooling for activity recognition. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1829–1837, 2018. 2

[38] Khurram Soomro, Amir Roshan Zamir, and M Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012. 5

[39] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Gate-shift networks for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1102–1111, 2020. 1, 2

[40] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2017. 2

[41] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 4041–4049, 2015. 2

[42] 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 *European conference on computer vision*, pages 20–36, 2016. 1, 2

[43] Yue Wang, Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang. E2-train: Training state-of-the-art cnns with over 80% energy savings. In *Advances in Neural Information Processing Systems*, pages 5138–5150, 2019. 2

[44] Simon Wiedemann, Temesgen Mehari, Kevin Kepp, and Wojciech Samek. Dithered backprop: A sparse and quantized backpropagation algorithm for more efficient deep neural network training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 720–721, 2020. 2

[45] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020. 2

[46] 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*, pages 6222–6231, 2019. 1, 2

[47] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *Advances in Neural Information Processing Systems*, pages 7780–7789, 2019. 2

[48] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis. Adaframe: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[49] 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*, pages 2678–2687, 2016. 1, 2

[50] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018. 2

[51] 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. 1, 2, 11, 12

Appendix

In addition to the comparison with 2D models, we also show results of our method compared to the state-of-the-art 3D models and additional 2D models on Breakfast, Something-Something V1 & V2, UCF-101 and HMDB51. [Nx] denotes the new citations in the tables.

Comparison on the Breakfast dataset.

Model	Backbone	#3D	#Optical flow	#Frames	#Clusters	Top-1
ResNet152[18]	ResNet152	-	-	64	-	41.1%
ActionVLAD [18]	ResNet152	-	-	64	-	55.5%
VideoGraph [18]	ResNet152	-	-	64	-	59.1%
TSM [29] (our impl.)	ResNet50	-	-	16	-	72.1%
I3D [18]	3D Inception-v1	✓	-	512	-	58.6%
I3D + ActionVLAD [18]	3D Inception-v1	✓	-	512	-	65.5%
I3D + VideoGraph [18]	3D Inception-v1	✓	-	512	-	69.5%
3D ResNet-50 + Timeception [17]	3D ResNet-50	✓	-	512	-	71.3%
Ours-slope	ResNet50	-	-	all	16	74.9%
Ours-cumulative	ResNet50	-	-	all	16	**76.6%**

Table 6. Our method using either slope temporal clustering or cumulative temporal clustering compared to existing works on the Breakfast dataset. Our proposal outperforms TSM and the 3D model, and significantly exceeds in top-1 accuracy methods using the deeper backbone architecture, ResNet-152. By using all frames our method has an advantage on long-term video action recognition.

Comparison on the Something-Something dataset.

Model	Backbone	#3D	#Optical flow	#Frames	#Clusters	Top-1 V1	Top-1 V2
TSN [29]	ResNet50	-	-	8	-	19.7%	30.0%
TRN-Multiscale [29]	ResNet50	-	-	8	-	38.9%	48.8%
TSM [29]	Resnet50	-	-	8	-	45.6%	59.1%
STM [22]	ResNet50	-	-	8	-	49.2%	62.3%
MSNet-R50 [1]	TSM-ResNet50	-	-	8	-	**50.9%**	**63.0%**
I3D [3]	I3D	✓	-	32	-	41.6%	-
NL-I3D [3]	I3D	✓	-	32	-	44.4%	-
NL-I3D+GCN [3]	I3D	✓	-	32	-	46.1%	-
S3D-G [4]	Inception	✓	-	64	-	48.2%	-
ECO [51]	BNIncep+3D Res18	✓	-	8	-	39.6%	-
ECO [51]	BNIncep+3D Res18	✓	-	16	-	41.4%	-
ECO-En *Lite* [51]	BNIncep+3D Res18	✓	-	92	-	46.4%	-
ECO-En *Lite*-RGB+Flow [51]	BNIncep+3D Res18	✓	✓	92+92	-	49.5%	-
DFB-Net [2]	3D ResNet50	✓	-	16	-	50.1%	-
Ours-slope	TSM-ResNet50	-	-	all	8	46.7%	60.2%
Ours-cumulative	TSM-ResNet50	-	-	all	8	49.5%	62.7%

Table 7. Top-1 accuracy on Something-Something V1 and V2 datasets. Our method using cumulative temporal clustering outperforms most state-of-the-art methods on both Something-Something V1 and V2, and performs on par with ECO-En *Lite* with both RGB and optical flow while slightly worse than MSNet-R50. Our method achieves limited accuracy improvement for shorter videos.

Comparison on the UCF-101 and HMDB51 dataset.

Model	Backbone	Pre-train	#3D	#Optical flow	#Frames	#Clusters	Top-1 UCF-101	Top-1 HMDB51
TSM [29] (our impl.)	ResNet50	Kinetics	-	-	1	-	91.2%	65.1%
TSN [29]	ResNet50	Kinetics	-	-	8	-	91.7%	64.7%
SI+DI+OF+DOF [1]	ResNeXt50	Imagenet	-	✓	dynamic images	-	95.0%	71.5%
TSM [29]	ResNet50	Kinetics	-	-	8	-	95.9%	73.5%
STM [22]	ResNet50	ImageNet+Kinetics	-	-	16	-	96.2%	72.2%
MSNet-R50 [1]	TSM-ResNet50	Kinetics	-	-	8	-	-	75.8%
ECO-En *Lite* [51]	BNIncep+3D Res18	Kinetics	✓	-	8	-	94.8%	72.4%
RGB I3D [2]	3D Inception-v1	Kinetics	✓	-	64	-	95.1%	74.3%
Two-stream I3D [2]	3D Inception-v1	Kinetics	✓	✓	64+64	-	**97.8%**	**80.9%**
Ours-slope	TSM-ResNet50	Kinetics	-	-	all	8	96.2%	73.3%
Ours-cumulative	TSM-ResNet50	Kinetics	-	-	all	8	96.4%	73.4%

Table 8. Top-1 accuracy on UCF-101 and HMDB51. Our method performs only slightly better than the state-of-the-art on the scene-related datasets UCF-101 and HMDB51, and worse than two-stream I3D, which uses both RGB and optical flow with 3D Inception-v1 backbone. Given that these datasets do not have a large number of frames per video, the improvement of our method over sampling methods is limited.

References

[1] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, 2020.

[2] Brais Martinez, Davide Modolo, Yuanjun Xiong, and Joseph Tighe. Action recognition with spatial-temporal discriminative filter banks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[3] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[4] 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)*, September 2018.