Dynamic Inference: A New Approach Toward Efficient Video Action Recognition

Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, Yi Yang, Shilei Wen
1 MMLab, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
2 Department of Computer Vision Technology (VIS), Baidu Inc., China
3 University of Chinese Academy of Sciences, China
4 University of Technology Sydney, Australia

{wh.wu,shifeng.chen}@siat.ac.cn {hedongliang01,tanxiao01,v.yangyi08,wenshilei}@baidu.com

Abstract

Though action recognition in videos has achieved great success recently, it remains a challenging task due to the massive computational cost. Designing lightweight networks is a possible solution, but it may degrade the recognition performance. In this paper, we innovatively propose a general dynamic inference idea to improve inference efficiency by leveraging the variation in the distinguishability of different videos. The dynamic inference approach can be achieved from aspects of the network depth and the number of input video frames, or even in a joint input-wise and network depth-wise manner. In a nutshell, we treat input frames and network depth of the computational graph as a 2-dimensional grid, and several checkpoints are placed on this grid in advance with a prediction module. The inference is carried out progressively on the grid by following some predefined route, whenever the inference process comes across a checkpoint, an early prediction can be made depending on whether the early stop criteria meets. For the proof-of-concept purpose, we instantiate several dynamic inference frameworks. In these instances, we overcome the drawback of limited temporal coverage resulted from an early prediction by a novel frame permutation scheme, and alleviate the conflict between progressive computation and video temporal relation modeling by introducing the online temporal shift module. Extensive experiments are conducted to thoroughly analyze the effectiveness of our ideas and to inspire future research efforts. Results on various datasets also evident the superiority of our approach.

1. Introduction

Action recognition in videos is one of the most active research topics in the computer vision community, owing to

* Work done during a research internship at Baidu.
† Corresponding author.
tentions are paid to the efficiency of CNN models.

There have already existed pretty much work on improving the efficiency of action recognition in videos [12, 35, 20, 2, 34, 27, 22, 28, 25, 19]. To reduce the computation cost (aka FLOPs, which is short for float-point operations), these works mainly focus on designing efficient network architectures to learn more representative features for classification. Meanwhile, all such solutions treat each video instance equally, i.e., all videos pass through a same CNN network route in the inference phase and a same number of frames are sampled from each video for testing, which we believe can be further improved by inference in a dynamic (or adaptive) fashion.

Our insight is that videos differentiate from each other in terms of their distinguishability, which leads to two consequences. First, varying number of frames are needed for recognizing videos. For instance, as illustrated in Fig.1(a), it is quite natural for a human to tell a video belongs to “Writing” when a very few frames at the beginning of a video are observed. However, to tell whether a video can be categorized into “Long Jump” or “Running”, we have to watch the video till its ending part (Fig.1(b) as an example). Second, varying network capability is needed for categorizing videos due to diverse dominance of their corresponding visual features. As shown in Fig.1(a), due to an irregular viewpoint, the second row is a non-typical “Writing” video so it demands more capable network for feature abstraction.

Motivated by these observations, in this paper, we propose a novel idea of dynamic inference to improve action recognition efficiency. Computation resource is adaptively allocated among different videos according to their distinguishabilities. The dynamic inference approach can be achieved from aspects of the network depth and the input video frames, or even in a joint input-wise and network depth-wise manner. Specifically, we treat input frames and network blocks as a 2-dimensional grid, where we have predefined K grid points as checkpoints, to each of which a prediction head is appended. At the inference phase, when ever the inference process comes across a checkpoint, an early prediction can be made depending on whether the early stop criteria meets. For the proof-of-concept purpose, dynamic inference frameworks are instantiated using the MSDNet [14] and ResNet [13] backbone CNNs. In these instances, we overcome the drawback of limited temporal coverage resulted from an early prediction by a novel frame permutation scheme and alleviate the conflict between progressive computation and video temporal relation modeling by introducing an online temporal shift module [20]. Extensive experiments are conducted on multiple well-known datasets, including Kinetics-400 [17], Something-Something v1 and v2 [9], UCF101 [24] and HMDB51 [18]. We empirically analyze the different behaviours of input-wise, network depth-wise as well as joint-input-depth-wise dynamic inferences and show their strength and weakness to readers for comprehensively understanding our idea. Besides, experimental results verify that our solution can significantly reduce the average FLOPs while maintaining excellent recognition accuracy, showing the superiority of dynamic inference for action recognition. Our major contributions are summarized as follows:

- We are the first to improve action recognition efficiency from the dynamic inference viewpoint, which is previously neglected. Our work makes the reader think of efficient action recognition differently from lightweight model designing.
- For proof of concept, we turn dynamic inference idea into practical network instances by proposing frame permutation and online temporal shift to tackle the raised obstacles of limited temporal coverage and conflict between progressive computation and temporal relation modeling.
- Extensive empirical analysis based on the three instances shows the strength and weakness of input-wise, network depth-wise and joint dynamic inferences under different circumstances. Our models also achieve significant efficiency improvement compared to their fixed inference counterparts.

2. Related Work

Action recognition has drawn great research attention in the community [29, 33, 23, 30, 6, 26, 1, 34, 22, 31, 27, 28, 35, 2, 12, 20]. Our work focuses on efficient action recognition and it is closely related to the following two lines of research jobs.

More efficient network architectures specifically designed for video recognition have been well studied in the literature. Following the I3D [1] paradigm for spatial-temporal modeling, S3D [34], P3D [22], R(2+1)D [27], MFNet [2] and StNet [12] are proposed to reduce computation overhead of 3D convolution while remaining the spatial-temporal modeling property. These works choose to decompose 3D convolution into 2D spatial convolution followed by 1D temporal convolution on either per convolution operation basis or per 3D convolution network block basis. There exist several other networks which merge 2D and 3D information in CNN blocks to enhance the feature abstraction capability and resort to shallower backbones for efficiencies, such as ECO [35] and ARTNet [28]. Another research direction is to superimpose motion information learning into appearance feature network to reduce the overhead of motion stream for two-stream solution [23], the design of the networks of such solutions is inspired by the calculation process of dense optical flow, typical works include optical flow guided feature network [25] and motion...
feature network proposed in [19]. All of these existing solutions can be regarded as orthogonal research directions to our dynamic inference idea, and they can be further improved with appropriate dynamic inference adaptation.

Dynamic network route has already been studied for the image classification task in the literature as well. In [11], depth dropout is used in the training phase for efficient deep residual network training. The idea of stochastic depth network [15] is also leveraged as sort of regularization of deep networks in the training phase. The dynamic network is also proposed for testing. [10] explored halting in recurrent networks to save computational cost. [14] propose to adaptively determine the network depth for different images and a multi-scale dense network is designed for image classification. Region level network depth adaptation [7] is also studied for image classification. [32] dynamically select which layers of a CNN should be skipped during inference. Network depth adaptation is a very effective solution for image classification, and significant inference efficiency improvement is achieved. Our work is largely inspired by these solutions for image recognition. However, as best as we can know, we are the first to leverage dynamic inference for video recognition so far.

3. Approach

3.1. Dynamic Inference Formulation

Dynamic inference for video recognition poses two orientations, namely input-wise and network depth-wise. Generally, action recognition can be formulated as a hypothesis of the previous predictions as stop criteria, and the early stop criteria is met when the stop criteria meets. Here, the criteria for the kth checkpoint is defined as the hypothesis of $C_k : \max \{s_k\} = T_k$, where $s_k$ is the classification score at the kth checkpoint and $T_k$ is a threshold. Fig.2(a) and Fig.2(b) illustrate the straightforward depth-wise and input-wise dynamic scheme, in which $E_i$ is implemented as 2D CNN block(s) to support progressive computation. In these cases, We can see naive depth-wise dynamic leaves temporal relation unexplored and input-wise dynamic suffers from limited temporal coverage of input frames when it stops early. To this end, we propose a frame permutation mechanism and leverage online temporal shift module to alleviate these issues.

\[
\begin{align*}
    &z_i = \begin{cases} 
    P_0(y_0) = P_0 \circ E_0(I_0), & \text{if } i = 0 \\
    P_i(y_i) = P_i \circ E_i(y_{i-1}, y_{i-2}, \ldots, y_0, I_i) & \text{if } i > 0 
    \end{cases}, \\
    &\begin{cases} 
    y_i = E_i(y_{i-1}, y_{i-2}, \ldots, y_0, I_i) = A_i(E_i(I_i), y_{i-1}, \ldots, y_0) \\
    I_i \neq None, \forall i \geq 0 
    \end{cases}
\end{align*}
\]

where $A_i$ is feature aggregation function (e.g., average pooling) and $F_0, ..., F_{i-1}$ are all the sampled frames and $i$ is the number of sampled frames.

From Eq.1, the following concerns on designing general dynamic inference networks raise naturally: 1) when early stop at $z_i$, the sampled input frames sets, $I_0, ..., I_i$, cover very limited part of the input video. 2) dynamic inference network should be enabled with progressive computation capability. The previous computation outputs should be utilized to support current computation incrementally. 3) temporal relation modeling is supposed to enhance recognition performance, meanwhile temporal modeling such as 3D convolution involves temporal dependencies and it conflicts with progressive computation requirement.

3.2. Instantiation

Following Eq.1, we can design a variety of dynamic inference frameworks. For the proof-of-concept purpose, we showcase how to instantiate dynamic frameworks by keeping the above three concerns in mind. We treat input frames sets (denoted as $I_0, I_1, ..., I_{N-1}$) and network blocks (denoted as $B_0, B_1, ..., B_{M-1}$) as a 2-dimensional grid, where we have predefined K grid points (denoted as $I_{0, j}, I_{1, j}, I_{2, j}, ..., I_{K-1, M}$) as checkpoints, to each of which a prediction head is appended. At the inference phase, whenever a checkpoint is reached, an early prediction can be made depending on whether the stop criteria is met. Here, the criteria for the kth checkpoint is implemented as 2D CNN block(s) to support progressive computation. In these cases, We can see naive depth-wise dynamic leaves temporal relation unexplored and input-wise dynamic suffers from limited temporal coverage of input frames when it stops early. To this end, we propose a frame permutation mechanism and leverage online temporal shift module to alleviate these issues.

3.2.1 Frame Permutation

Frame permutation scheme can enlarge the temporal coverage of input frames when it stops early and construct diverse temporal strides for input. To exemplify this, suppose eight frames are evenly sampled from a video with a length of $L$, we can see from Fig.2(b), if three frames are processed and early stop is made, the temporal coverage of the in-
Figure 2. Illustration of different dynamic inference schemes when five blocks and eight frames are used for recognizing a video. Blue arrows and circles denote 2D-CNN blocks and their corresponding feature maps. Green bricks mean prediction heads and yellow arrows mean online temporal shift modules. The solid blue circles represent predefined checkpoints in our framework. (a) and (b) show a naive 5-level depth-wise and 4-level input-wise dynamic inference solution, respectively. (c) depicts that with frame shuffle, each group of input frames constructs an input sequence with multiple temporal strides and the temporal coverage of the valid input is enlarged if early stop is made. (d) shows our proposed frame permutation + online temporal shift solution.

In this paper, we set \( N \) to 8. We start with the case of \( L = 8 \). The extract frames \([F_0, F_1, ..., F_7]\) are permuted such that

\[
[\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_7] = [F_0, F_3, F_5, F_6, F_1, F_4, F_2, F_7].
\]

The temporal order of the shuffled frames no longer holds, and motion noise will be introduced. We divide them into two groups, \([F_0, F_3, F_5, F_6]\) and \([F_1, F_2, F_4, F_7]\), such that frames in both groups keep in sequential order. We can also see there are multiple temporal strides, namely \(3L/2N, 2L/8\) in both groups. If temporal relation is modeled inside each group, multiple strides can help to capture better temporal dynamics of different actions whose motion intensities are varying. When the number of sampled frames \( l \) equals \( N \times E \) where \( E \) is the size of each frames set, then our permutation mechanism designed for eight frames can be extended as follows. Firstly, the sampled frames are divided into eight sets

\[
\mathcal{F}_i = \{F_{iE}, F_{iE+1}, ..., F_{(i+1)E-1}\}, i = 0, 1, ..., 7.
\]

Then, we shuffle the input frames at frame-set granularity

\[
[\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_N] = [\mathcal{F}_0, \mathcal{F}_3, \mathcal{F}_5, \mathcal{F}_6, \mathcal{F}_1, \mathcal{F}_4, \mathcal{F}_2, \mathcal{F}_7].
\]

The shuffled frame sets are also divided into two groups. Correspondingly, each block of the computation graph is extended to \( E \) copies for processing \( E \) frames in a set \( \mathcal{F}_i \).

3.2.2 Online Temporal Shift

To tackle the problem of missing temporal relation when only 2D convolutions are applied on input frames, we adopt a temporal shift module for efficient temporal modeling. The original temporal shift module [20] shifts part of the channels from feature maps of each input frame forward and backward to that of its adjacent input frames, which facilitates information exchange among neighboring frames with zero FLOPs cost and negligible time cost. Under dynamic inference circumstance, we tailor original TSM approach by only applying forward shift, which makes it possible for progressive computation. Specifically, we add an online temporal shift module between \( \mathcal{I}_mB_m \) and \( \mathcal{I}_{m+1}B_m \) for arbitrary \( n \) and \( m \) to fully take advantage of multi-stride input brought by the frame permutation. The overall unified framework is shown in Fig.2(d). It is worthy of noting that, our method is a general design, when the \( K \) checkpoints are located in the last row or last column, it is specialized to be input-wise or depth-wise dynamic inference framework.

4. Experiments

To validate effectiveness of our idea and provide comprehensive understanding on dynamic inference, we implemented three frameworks based on state-of-the-art 2D CNN MSDNet-38, ResNet-50 and ResNet-101, respectively. More deeper and powerful backbones could be leveraged, but here we do not show such experiments because 1) we focus on efficient action recognition and heavy network is not suitable; 2) the three instances are able to achieve proof of concept purpose. MSDNet-38 contains five blocks (M=5) and four scales (S=4) are used in each block. Note that the online temporal shift module is only adopted on feature maps at the coarsest scale for all layers except the first layer. Fig. 3 illustrated the framework using only three scales for convenience. \( K \) is 6 for MSDNet and each prediction head is composed of an average pooling and a linear layer. For ResNet, online temporal shift model is added at
each residual block and we only append early-exit classifiers to the Res3, Res4 and Res5 and $K$ is set to 4 (the last 4 checkpoints in Fig. 2(d)). Each classifier has two down-sampling convolution layers with $3 \times 3$ filters, followed by an average pooling and a linear layer. For $T_k$, we use the same implementation details as in [14].

### 4.1. Datasets and Evaluation Metrics

To comprehensively evaluate our proposed method, we perform extensive experiments on the recent large scale dataset named Kinetics-400 [17]. We also conduct experiments on heavy temporal relation sensitive datasets, including Something-Something v1 & v2 [9]. For these datasets, the actions therein mainly include object-object and human-object interactions, which require strong temporal relation to well categorizing them. Moreover, transfer learning experiments on the UCF-101 [24] and HMDB-51 [18], which are much smaller than Kinetics-400, is carried out to show the transfer capability of our solution. The evaluation metric is top-1 precision for Kinetics-400 and Something-Something. We also report average FLOPs/Video in the testing phase as well as the number of model parameters to depict model complexity. In this paper, we only use the RGB frames of these datasets for experiments.

### 4.2. Implementation Details

#### Training

Data augmentation and preprocessing strategy is the same as TSN [30]. In our experiments, $l$ is set to 16. To evaluate the proposed network on several action recognition datasets, we pre-trained MSDNet-38, ResNet-50, ResNet-101 on the ImageNet-1k [3] for initialization. The loss is the sum of cross-entropy at each checkpoint. For Kinetics-400, Something-Something v1 & v2, we start training with a learning rate of 0.01 and reduce it by a factor of 10 at 25, 35, 45 epochs and stop at 50 epochs. Dropout ratio is 0.5. Since UCF-101 and HMDB-51 are not large enough and are prone to over-fitting, we followed the common practice to use Kinetics pre-trained model as initialization and in total 25 epochs are trained, with an initial learning rate of 0.01 and it is decayed by 10x every ten epochs. Higher dropout ratio of 0.8 is used. Stochastic gradient descent (SGD) with a mini-batch size of 128 is utilized as an optimizer, and its momentum and weight decay value is set to 0.9 and 5e-4, respectively. BatchNorm layers [16] are all finetuned during training.

#### Inference

We apply single centre 224x224 cropping to predict the class labels and the 16 frames are evenly sampled from videos. To get the accuracy-FLOPs curves in the paper, we traverse different budget $Q$ to determine $T_k$ on the train set and measure the actual accuracy and FLOPs. As for validation set, we use the critical point of $Q^*$, where accuracy drops little and FLOPs are smaller than full inference, obtained from the accuracy-FLOPs curve on the validation set to determine $T^*_k$ for testing and then accuracy and average FLOPs per video is calculated.

### 4.3. Empirical Analysis

**MSDNet vs. ResNet**

First of all, we would like to know how backbone makes a difference. Both Kinetics-400 and Something-Something v2 is used to evaluate the dynamic inference mechanisms proposed in Section 3 using the RGB modality. As shown in Fig. 4(a), MSDNet-38 performs better than ResNet-50 in depth-wise dynamic inference owing to the multi-scale feature maps and dense connectivity. Influenced by the limited performance of depth-wise dynamic inference, joint depth-wise and network depth-wise solution performed worse than input-wise dynamic inference for ResNet-50. On Something-Something v2, similar observations can be found.

**Ablation studies**

Fig. 5 and Fig. 6 show the results based on MSDNet under different dynamic inference settings. Accordingly, the effectiveness of each component in our framework is analyzed as follows.

**Depth-wise dynamic inference**

As shown in Fig. 5 and Fig. 6, it can be concluded from the results that depth-wise
Dynamic inference is effective on both datasets no matter whether the online temporal shift is enabled or not. It is a very interesting observation from the results of Fig. 6(b) that depth-wise dynamic inference with online temporal shift on Something-Something v2 performs even slightly better than joint input-wise and depth-wise adaptation. This is reasonable because temporal relation is crucial for this dataset, input adaptation brings very little gain such that a network trained with pure depth-wise adaptation can be more competitive than its joint depth-wise and input-wise version. We can see that on Something-Something v2, without and with the online temporal shift, depth-wise adaptation achieves top-1 accuracy of 31.75% and 60.02%, and the #FLOPs can be reduced to 44.6G and 35.4G, respectively.

Input-wise dynamic inference and frame permutation We find that on Something-Something v2, input-wise adaptation brings negligible FLOPs saving, while on Kinetics-400 it can save FLOPs to some extent. In Fig. 5 and Fig. 6, on both datasets, it is clear that frame permutation improves input-wise adaptation, regardless of whether online temporal shift module is used or not. Even though temporal relation is not so crucial on Kinetics-400 compared to Something-Something v2, frame permutation still gains a lot, and we can infer that temporal coverage of the input sequence is essential to video classification. Naturally, other architectures such as TSM, can also benefit from the frame permutation mechanism for dynamic inference.

Online temporal shift Temporal modeling by online temporal shift is also essential for the final performance. It is seen from the figures, the online temporal shift improves the top-1 accuracy upper bound of our framework from 69.32% to 71.17% and 31.64% to 59.80% on Kinetics-400 in Fig. 5 and Something-Something v2 in Fig. 6, respectively.

Joint input-wise and depth-wise dynamic inference On Kinetics, the proposed joint solution benefits from an adaption of both dimensions in Fig. 5. The #FLOPs is reduced to 24.3G and 26.5G from 52G without or with the online temporal shift, respectively. However, in Fig. 6, on Something-Something v2, it performs comparably with or even slightly worse than depth-wise adaptation owing to the heavy temporal dependency characteristic of the dataset and input-wise adaptation can hardly bring any gain.

4.4. Comparison with State-of-the-arts

Results on Kinetics-400 We evaluate the proposed framework against the recent state-of-the-art 2D/3D convolution-based solutions. Extensive experiments are conducted on the Kinetics-400 to compare all models in terms of their effectiveness (i.e., top-1 accuracy) and efficiency (reflected by the total number of model parameters and FLOPs needed in the inference phase). Results are summarized in Table 1. To be noted, the latest camera ready version of TSM reported results of these datasets using dense sampling for higher performance. Hence, in this paper we cited the results of these datasets using uniform sampling reported in its arXiv version.

The results show that our proposed dynamic inference with MSDNet-38 backbone achieves top-1 precision of 71.20% with single RGB modality, and it requires only 26.5G FLOPs per video on average. When large backbone models are used, our method still strikes good performance-FLOPs trade-off. Compared to StNet-Res101, our method with ResNet-50 exhibits better recognition performance (72.57% vs. 71.38%), but the #FLOPs are significantly reduced by over 10× (from 310G to 35G). When compared to the recent state-of-the-art ECO and TSM-Res50, our model improves the accuracy from 69% and 72.5% to 72.57%, meanwhile the #FLOPs is reduced from 64G and 64G to 35G, respectively. Other methods, such as MF-Net, S3D, and R(2+1)D, though their marginal performance gains are obtained at the price of at least 10 times of computation cost, our method with ResNet-101 still achieve better recognition performance than them while #FLOPs is only 66G. In brief, our model achieves excellent performance-cost trade-off with different backbones.

We also show the distribution of videos predicted at each checkpoint in Table 2. More results and visualizations are

1https://arxiv.org/abs/1811.08383v1
<table>
<thead>
<tr>
<th>Framework</th>
<th>Backbone</th>
<th>Input × # Clips</th>
<th>Prec@1</th>
<th># Params</th>
<th>FLOPs/Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>13D [1]</td>
<td>3D BN-Inception</td>
<td>[All×3×256×256]×1</td>
<td>70.24</td>
<td>12.7M</td>
<td>544.44G</td>
</tr>
<tr>
<td>S3D [34]</td>
<td>3D BN-Inception</td>
<td>[All×3×224×224]×1</td>
<td>72.20</td>
<td>8.8M</td>
<td>518.66G</td>
</tr>
<tr>
<td>ARTNet with TSN [28]</td>
<td>3D ResNet-18</td>
<td>[16×3×112×112]×250</td>
<td>69.2</td>
<td>35.2M</td>
<td>5925G</td>
</tr>
<tr>
<td>MF-Net [2]</td>
<td>-</td>
<td>[16×3×224×224]×1</td>
<td>65.00</td>
<td>8.0M</td>
<td>11.1G</td>
</tr>
<tr>
<td>ECO [35]</td>
<td>BN-Inception+3D ResNet-18</td>
<td>[16×3×224×224]×1</td>
<td>69.00</td>
<td>47.5M</td>
<td>64G</td>
</tr>
<tr>
<td>R(2+1)D RGB [27]</td>
<td>ResNet-34</td>
<td>[32×3×112×112]×10</td>
<td>72.00</td>
<td>63.8M</td>
<td>1524G</td>
</tr>
<tr>
<td>Nonlocal-I3d [31]</td>
<td>ResNet-50</td>
<td>[128×3×224×224]×1</td>
<td>67.30</td>
<td>35.33M</td>
<td>145.7G</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>[8×3×224×224]×1</td>
<td>66.80</td>
<td>24.3M</td>
<td>33G</td>
<td></td>
</tr>
<tr>
<td>ResNet-50</td>
<td>[16×3×224×224]×1</td>
<td>67.80</td>
<td>24.3M</td>
<td>64G</td>
<td></td>
</tr>
<tr>
<td>StNet [12]</td>
<td>ResNet-50</td>
<td>[16×3×224×224]×1</td>
<td>72.50</td>
<td>24.3M</td>
<td>64G</td>
</tr>
<tr>
<td>Proposed</td>
<td>MSDNet-38 (Full)</td>
<td>[25×15×256×256]×1</td>
<td>69.85</td>
<td>33.16M</td>
<td>189.29G</td>
</tr>
<tr>
<td></td>
<td>MSDNet-38</td>
<td>[25×15×256×256]×1</td>
<td>71.38</td>
<td>52.15M</td>
<td>310.50G</td>
</tr>
<tr>
<td></td>
<td>ResNet-50</td>
<td>[25×15×224×224]×1</td>
<td>71.17</td>
<td>62.31M</td>
<td>52G</td>
</tr>
<tr>
<td></td>
<td>ResNet-50</td>
<td>[16×3×224×224]×1</td>
<td>72.57</td>
<td>29.12M</td>
<td>35G</td>
</tr>
<tr>
<td></td>
<td>ResNet-101</td>
<td>[16×3×224×224]×1</td>
<td>74.70</td>
<td>48.12M</td>
<td>66G</td>
</tr>
</tbody>
</table>

Table 1. Comparison of our method with several state-of-the-art 2D/3D convolution-based solutions. The results are reported on the validation set of Kinetics-400, with RGB modality only. We investigate both Prec@1 and model efficiency w.r.t. the total number of model parameters and FLOPs needed in inference. Here, “All” denotes using all frames in a video.

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Videos</td>
<td>1993</td>
<td>2392</td>
<td>2871</td>
<td>3445</td>
<td>4134</td>
<td>4961</td>
<td>19796</td>
</tr>
</tbody>
</table>

Table 2. Number of videos which stop at each checkpoint on Kinetics-400 validation set.

Results on Something-Something v1 & v2: The Something-Something datasets are more complicated than Kinetics, the comparison of our solution against existing state-of-the-arts are list in Table 3. It is observed that, with 16 frames as input, our solution achieves state-of-the-art top-1 accuracy of 46.5% on Something-Something v1 validation set and 60.02% on v2. The #FLOPs of our model is 38.4G/35.4G on v1 and v2 respectively, which is much smaller than ECO ENLite, Non-local I3D+GCN and TSM16F. Compared to TSM on both datasets, our method achieves better recognition performance than TSM16F while the #FLOPs is comparable with TSM8F. We have also submitted the testing results of Something-Something v2 and results show that the testing performance of our solution is as high as 60.16% in terms of top-1 accuracy.

4.5. Transfer Learning on UCF-101 & HMDB-51

We transfer the proposed models pre-trained on Kinetics-400 to the much smaller datasets of UCF-101 and HMDB-51 to show that our method can be well generalized to other datasets. Fig. 7 show the Mean Class Accuracy vs. Average FLOPs/Video curve on the split1 of UCF-101 and HMDB-51, respectively. For UCF-101 split1, the average FLOPs can be reduced by 70% while remaining the same performance with full inference. It is also be observed with 1% accuracy degradation, around 80% reduction in average FLOPs is achieved. For HMDB-51, our dynamic inference scheme keep classification accuracy while reducing up to 45% average FLOPs. Besides, the accuracy on UCF-101 is much higher than HMDB-51, and we believe that our method can reduce more average FLOPs on easier dataset. We also observe that the curve of HMDB-51 is similar to that of Kinetics-400 because the accuracy of our model is
Table 3. Performance and FLOPs consumptions of our method on the Something-Something v1 and v2 datasets compared with the state-of-the-art methods. FLOPs/Video are averaged over all the videos from the validation set.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Pretrain</th>
<th>FLOPs/Video</th>
<th>Something-Something v1</th>
<th>Something-Something v2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>top-1 test</td>
<td>top-5 test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>top-1 val</td>
<td>top-5 val</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARTNet with TSN</td>
<td>3D ResNet-18</td>
<td>5925G</td>
<td>94.3</td>
<td>70.9</td>
<td></td>
</tr>
<tr>
<td>ECO</td>
<td>BNInception+</td>
<td>64G</td>
<td>92.8</td>
<td>68.5</td>
<td></td>
</tr>
<tr>
<td>3D RGB</td>
<td>3D ResNet-18</td>
<td>64G</td>
<td>92.8</td>
<td>68.5</td>
<td></td>
</tr>
<tr>
<td>TSN RGB</td>
<td>BNInception</td>
<td>33G</td>
<td>91.1</td>
<td>63.2</td>
<td></td>
</tr>
<tr>
<td>TSN 16F</td>
<td>ResNet-50</td>
<td>33G</td>
<td>91.5</td>
<td>63.6</td>
<td></td>
</tr>
<tr>
<td>TSN 50F</td>
<td>ResNet-50</td>
<td>33G</td>
<td>94.0</td>
<td>70.3</td>
<td></td>
</tr>
<tr>
<td>TSM 50F</td>
<td>ResNet-50</td>
<td>33G</td>
<td>94.0</td>
<td>70.3</td>
<td></td>
</tr>
<tr>
<td>TSM 50F</td>
<td>ResNet-50</td>
<td>64G</td>
<td>94.5</td>
<td>70.7</td>
<td></td>
</tr>
<tr>
<td>SiNet</td>
<td>MSDNet-38</td>
<td>15.8G</td>
<td>94.2</td>
<td>-</td>
<td>70.1</td>
</tr>
<tr>
<td></td>
<td>ResNet-101</td>
<td>18.5G</td>
<td>94.7</td>
<td>-</td>
<td>72.34</td>
</tr>
<tr>
<td></td>
<td>MSDNet-38</td>
<td>34.4G</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>MSDNet-38</td>
<td>34.6G</td>
<td>95.3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResNet-101</td>
<td>69.1G</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Transfer learning performances with RGB modality. Note that the #FLOPs herein takes into account the testing strategies, such as multi-crop testing, for different models.

very close on these two scene-based datasets.

When compared with other methods, our model is evaluated by following the common practice of averaging accuracy/FLOPs over three training/testing splits of both datasets. The evaluation results can be found in Table 4. From this table, we can see that our model shows a pretty transfer capability. It obtains pretty good accuracy on UCF-101 and HMDB-51, and the mean class accuracy is 94.2% and 70.1%, respectively, which is comparable to or even better than performances of many state-of-the-art solutions. Besides, our model achieves great efficiency improvement, and the average FLOPs/Video can be significantly reduced to 15.8G and 29.2G on UCF-101 and HMDB-51. The results of ResNet based models can be analyzed similarly.

5. Discussion

Parallelism Dynamic inference, compared with fixed inference, could degrade the parallelism capability to some extent, but it can be paralleled. The depth-wise dynamic inference can be paralleled in the input dimension, meanwhile input-wise dynamic inference can be paralleled in the input dimension when the computation process is going from one checkpoint to the next one and more than one frames are fed. As for the joint input-wise and depth-wise scheme, it’s parallelism capability is further limited, take Fig.2(d) for illustration, if the computation process goes from the 4th checkpoint to the 5th one, the last block outputs of \([F_0, F_3, F_5, F_6]\) can be calculated in parallel. When it travels from the 5th checkpoint to the last one, the whole feature extraction process of \([F_1, F_2, F_4, F_7]\) can be paralleled.

Optimality Our key insight is dynamic inference and we formally summarize its general idea and provide instantiating strategies to prove its feasibility. These framework instances are designed to thoroughly analyze effectiveness of our idea. However, the optimality of these frameworks are not necessarily guaranteed. For example, how these thresholds \(T_k\) can be adaptively determined for different videos at different checkpoints; how the checkpoints can be dynamically located from video to video and so on. We believe that all of these problems deserve future research effort.

6. Conclusion

In this paper, we focus on improving action recognition efficiency in videos by dynamic inference, which takes advantage of distinguishability variation among different videos. By exploiting dynamic inference, our models are verified on multiple well-known datasets being able to significantly save FLOPs consumption while remaining recognition performance unaffected. To our best knowledge, we are the first to propose dynamic inference for efficient video recognition, and it is proven to be a promising direction.

Acknowledgment The work was supported by Key-Area Research and Development Program of Guangdong Province (2019B010155003), National Natural Science Foundation of China (U1713203), and Shenzhen Science and Technology Innovation Commission (Project KQJSCX20180330170238897).
References


[34] 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. 2, 7