2D or not 2D? Adaptive 3D Convolution Selection for Efficient Video Recognition

Hengduo Li[1] Zuxuan Wu[2*] Abhinav Shrivastava[1] Larry S. Davis[1]
[1] University of Maryland [2] Fudan University
{hdli,abhinav,lsd}@cs.umd.edu zxwu@fudan.edu.cn

Abstract

3D convolutional networks are prevalent for video recognition. While achieving excellent recognition performance on standard benchmarks, they operate on a sequence of frames with 3D convolutions and thus are computationally demanding. Exploiting large variations among different videos, we introduce Ada3D, a conditional computation framework that learns instance-specific 3D usage policies to determine frames and convolution layers to be used in a 3D network. These policies are derived with a two-head lightweight selection network conditioned on each input video clip. Then, only frames and convolutions that are selected by the selection network are used in the 3D model to generate predictions. The selection network is optimized with policy gradient methods to maximize a reward that encourages making correct predictions with limited computation. We conduct experiments on three video recognition benchmarks and demonstrate that our method achieves similar accuracies to state-of-the-art 3D models while requiring $20\% - 50\%$ *less computation across different datasets. We also show that learned policies are transferable and Ada3D is compatible to different backbones and modern clip selection approaches. Our qualitative analysis indicates that our method allocates fewer 3D convolutions and frames for "static" inputs, yet uses more for motion-intensive clips.*

1. Introduction

Videos are expected to make up a staggering 82% of Internet traffic by 2022 [1], which demands approaches that can understand video content like actions and events accurately and efficiently. Key to video recognition is temporal modeling to capture relationships among different frames. Towards this goal, extensive studies have been conducted with 3D convolutional networks by extending 2D convolutions over time [33, 4, 35, 34, 9, 8, 45]. While offering excellent recognition accuracy on standard benchmarks [4, 13, 16], 3D models are often computationally expensive due to the costly convolution operations along the temporal axis on a large number of stacked frames. For example, at the clip-level [1], a standard ResNet50 [14] model only requires 4.1 GFLOPs (giga floating-point operations) to compute predictions for a single image, while a SlowFast network [9] with the same ResNet50 backbone needs 16 times more computation (65.7 GFLOPs). Furthermore, the computational cost linearly grows with the number of clips uniformly sampled through the entire sequence for video-level prediction aggregation.

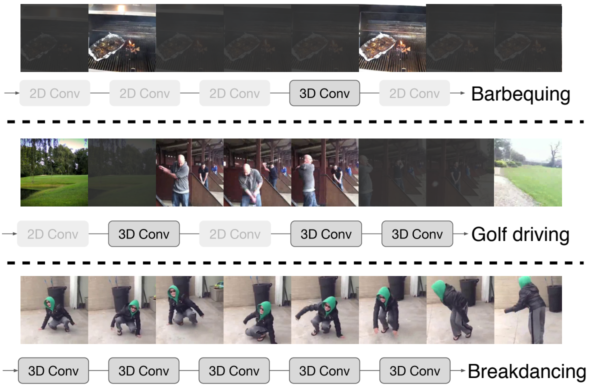

Figure 1: **A conceptual overview of our approach**. Ada3D learns to adaptively keep/discard 3D convolutional layers and frames conditioned on input clips for efficient video recognition. Fewer 3D convolutions and frames are kept for clips that contain discriminative static cues and contextual information, while more are used for motion-intensive clips, in pursuit of a reduced overall computational cost without sacrificing recognition accuracy. Black mask indicates the frame is discarded.

But are 3D convolutions really important for recognizing different types of videos? Do we really need them through-

[*] Corresponding author.

[1] Here, we use "clip" in a broad sense; for 2D models, a clip is a single RGB frame while for 3D models it is a stack of frames.

out the network? Is it necessary to perform 3D convolution on a fixed number of stacked frames for all different samples? Intuitively, 3D convolutions are critical for capturing changing patterns among inputs. However, due to large intra-class and inter-class variations, some videos are relatively more "static" than others, for which using a computationally expensive 3D model on redundant inputs might be unnecessary. This paper seeks to develop a computationally efficient framework for video recognition by learning how many frames to use and whether to use 3D convolutions in 3D networks. This is an orthogonal yet complementary direction to existing work on fast video recognition, which either designs lightweight 3D architectures [35, 44, 8, 34] or develops clip selection schemes to use fewer clips for classification [43, 20, 12, 41, 48].

With this in mind, we introduce Ada3D, an end-to-end framework that learns adaptive 3D convolution usage conditioned on each input clip sample for efficient video recognition. For each clip, deriving a dynamic inference strategy entails (1) learning how many frames are used as inputs to the 3D network; (2) conditioned on these selected frames, determining how many 3D convolutional layers are activated; (3) and most importantly, making correct predictions while only using a small number of input frames and 3D convolutions. By doing so, Ada3D allocates more computational resources to videos with complicated motion patterns while performing economical inference for "easy static" videos, enabling efficient video classification while maintaining reliable classification accuracy. While appealing, learning whether to keep/discard input frames and 3D convolutions is a non-trivial task, as it requires making *binary* decisions that are non-differentiable.

To this end, Ada3D is built upon a reinforcement learning framework [32]. In particular, given a video clip, Ada3D trains a two-head selection network to produce a frame usage policy and a convolution usage policy, indicating which frames in the input stack and which 3D convolutions in the network should be kept or discarded, respectively. Then, conditioned on the derived policies, dynamic inference is performed on a pretrained 3D network with selected frames and 3D convolutions for fast recognition. The selection network is optimized with policy gradient methods [32] to maximize a reward function that is carefully designed to incentivize using as few computational resources as possible while making correct predictions. We further jointly finetune the selection network with the 3D network such that the 3D model is able to adapt to the adaptive inference paradigm. It worth nothing that the selection network is designed to be lightweight so that its computational overhead is negligible.

We conduct extensive experiments to evaluate Ada3D on ActivityNet [16], FCVID [19], Mini-Kinetics-200 [44, 4], and demonstrate that Ada3D is able to save 20% to 50% computation on different datasets while maintaining similar recognition performance compared with baselines. We show policies learned on Mini-Kinetics-200 can be further transferred to the full Kinetics dataset [4]. In addition, we show the approach is compatible with different 3D models and it is also complementary to other clip-level selection methods [20, 43, 41, 12, 48]. We also demonstrate qualitatively that our method learns to allocate fewer 3D convolutions and frames for clips that are relatively more static, while applying more computation to motion-intensive clips.

2. Related Work

Deep neural networks for video recognition. Existing work typically designs video recognition architectures by equipping state-of-the-art 2D models with the ability for temporal modeling, and can be roughly categorized into two directions. In particular, the first applies 2D models on a per-frame basis and then model temporal relationships across frames by aggregating features along the temporal axis with operations such as pooling [38, 31, 10], recurrent networks [6, 47, 23], and using inputs with explicit temporal information such as optical flow [31, 10, 38]. The other [4, 33, 29, 35, 9, 8] directly transforms 2D models into 3D models with 3D convolutions applied on stacked RGB frames (clips). While achieving state-of-the-art performance on various benchmarks [4, 16, 13], 3D models are computationally expensive, limiting their deployment in real-world applications with limited resources. Our work aims to reduce the computational cost of 3D models by learning instance-specific 3D policies using fewer frames and 3D convolutions in a 3D model conditioned on inputs while making correct predictions at the same time.

Efficient video recognition. Extensive studies have been conducted on designing efficient network architectures for video recognition [50, 5, 35, 8, 49, 24, 34]. Recent advances in efficient 2D ConvNets, *e.g.* group convolution [17, 30], have been explored in 3D models [5, 35, 34]. In addition, some lightweight temporal aggregation operations are introduced to speed up inference such as a relational module in TRN [49] and a shift module in TSM [24]. More recently, X3D [8] expands a tiny model across several dimensions for a good efficiency/accuracy trade-off. However, all these approaches use a *fixed* input sampling scheme (*i.e.*, number of frames and frame rate) and compute predictions with a "one-size-fits-all" model for all inputs clips, regardless of the large temporal variations among them. In contrast, we learn dynamic frame usage policies and convolution usage policies conditioned on input clips, in pursuit of computational efficiency without sacrificing accuracy. It is worth pointing out that our method is model-agnostic, and can be used in tandem with these efficient networks.

Adaptive computation. Many adaptive computation

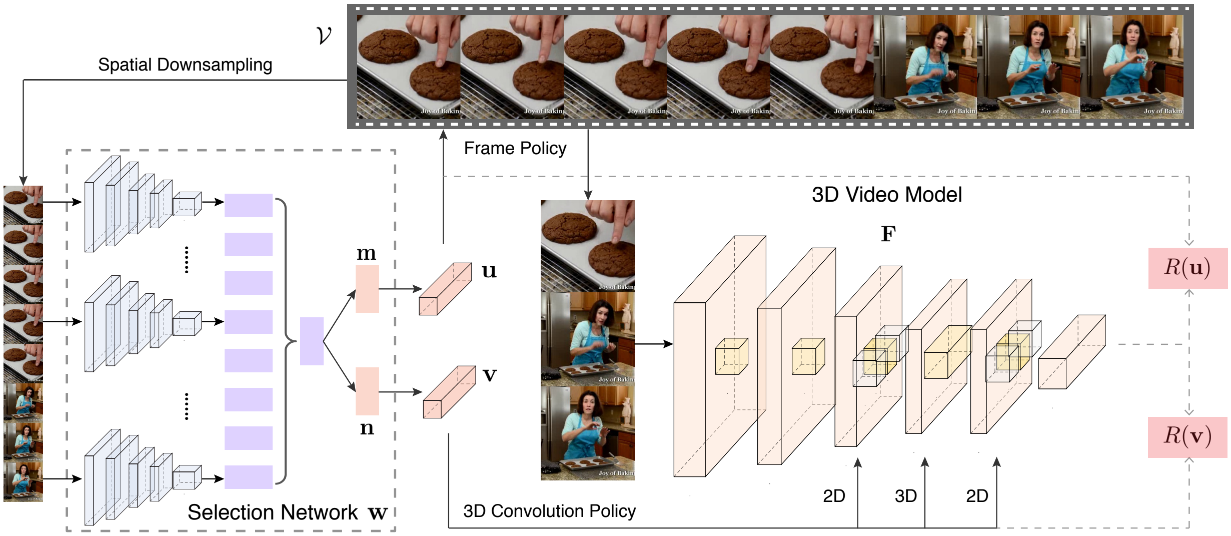

Figure 2: **An overview of our approach.** Given an input clip, the selection network produces features for each frame in the clip, which are further aggregated uniformly to derive a frame usage policy and a convolution usage policy simultaneously. These policies activate a subset of frames and 3D convolutions in the 3D network for inference. Then, conditioned on the prediction, two rewards are computed to evaluate the frame and convolution policy, respectively. See texts for more details.

(*a.k.a*, conditional computation) methods have been developed in the image domain, achieving reduced computation by dynamically selecting channels [2, 25], skipping layers [42, 11, 40, 37], performing early exiting with auxiliary structures [22, 18, 3, 46], adaptively switching input resolutions [27, 36, 46], *etc*. There are also a few recent studies exploring adaptive computation for videos. These approaches adaptively select salient clips for faster inference with one [43] or more [41] agents to aggregate video-level predictions. Compressed video [20] and audio [12, 20] are also utilized for further improvement in clip selection. More recently, a dynamic resolution selection strategy is introduced in [26].

Our method is closely related yet orthogonal to these approaches. They focus on selecting informative clips throughout the entire sequence to achieve fast inference, aiming to improve the widely used uniform sampling baseline for video recognition. For each selected clip, the same amount of computational resource is used. In contrast, we allocate computation conditioned on the complexity of the input video clip. This can be considered as dynamic routing in a network and is complementary to those clip-selection methods (as will be shown empirically) [43, 12, 20], which are a form of routing across different time steps in videos.

3. Approach

Ada3D reduces the computational cost of 3D networks by learning instance-specific 3D usage policies that encourage using fewer computational resources, in the forms of frames and 3D convolutions, while producing accurate predictions. To this end, we first revisit popular 3D networks used for temporal modeling in Sec. 3.1, and then elaborate different components of Ada3D in Sec. 3.2

3.1. 3D Networks for Video Recognition

Operating on stacked RGB frames, 3D video models typically extend state-of-the-art 2D networks by replacing a number of 2D convolutions with 3D convolutions for temporal modeling over time. Formally, taking as inputs an input clip $\mathcal{V}$ with T frames $\{v_1, v_2, ..., v_T\}$, 3D models obtain final predictions through a stack of 2D ($k_{1\times d\times d}$) and 3D ($k_{t\times d\times d}$) convolutional layers, where t denotes the temporal extent of 3D convolutional filters which is typically set to 3 and 5 in practice, and d denotes the spatial height and width. In common instantiations of 3D video models [33, 35, 4, 44, 9, 8], 3D convolutions are inserted into the building blocks of 2D networks, and these 3D blocks are organized based on heuristics such as using them in early [44, 35] or late [9, 44, 45] stages of the network, if not applied in all stages [4, 8, 35, 33]. Note that state-of-the-art frameworks usually perform temporal convolutions in a non-degenerate form [9, 8], *i.e*., taking in T frames and outputting T convolved frames. While achieving state-of-the-art recognition performance, 3D video models are often computationally expensive since a number of costly 3D convolutions are applied on a sequence of stacked frames.

3.2. Ada3D: Adaptive 3D Convolution Selection

Ada3D learns 3D convolution usage policies conditioned on input video clips to reduce the computational cost of 3D models. We achieve this with a lightweight selection network that is trained to determine which frames to use as inputs to a pretrained 3D model and which convolution layers to activate in the network for those selected frames. This involves making binary decisions that are non-differentiable, and thus not applicable for supervised frameworks. Instead, we formulate learning the selection network as Markov Decision Process (MDP) [28]. We define the state space of the MDP as the input video clip; actions in the model involve keeping/discarding frames and 3D convolutions in 3D networks. The reward balances between recognition accuracy and computation. The MDP is single-step: a video clip is observed, actions are taken, and a reward is computed—this can also be considered as a contextual bandit [21].

More formally, given an input clip $\mathcal{V}$ of length T and a 3D ResNet video classifier $\mathbf{F}$ with K 3D convolution stages[2], the selection network f_p, parameterized by $\mathbf{w}$, computes features for each frame in the input clip; these features are then aggregated as inputs to two parallel branches, outputting two vectors $\mathbf{m} \in \mathbb{R}^T$ and $\mathbf{n} \in \mathbb{R}^K$:

$$\mathbf{m}, \mathbf{n} = \texttt{sigmoid}(f_p(\mathcal{V}; \mathbf{w})). \tag{1}$$

Here, each entry in $\mathbf{m}$ and $\mathbf{n}$ is normalized to be in the range $[0, 1]$ with the $\texttt{sigmoid}(x) = \frac{1}{1+\exp(-x)}$ function, indicating the likelihood of keeping the corresponding frame and 3D convolution stage [2].

We then define a frame usage policy π^f and a convolution usage policy π^c with a T-dimensional and a K-dimensional Bernoulli distribution, respectively:

$$\pi^f(\mathbf{u} \,|\, \mathcal{V}) = \prod_{t=1}^{T} \mathbf{m}_t^{\mathbf{u}_t} (1 - \mathbf{m}_t)^{1-\mathbf{u}_t} \tag{2}$$

$$\pi^c(\mathbf{v} \,|\, \mathcal{V}) = \prod_{k=1}^{K} \mathbf{n}_k^{\mathbf{v}_k} (1 - \mathbf{n}_k)^{1-\mathbf{v}_k}. \tag{3}$$

where $\mathbf{u} \in \{0, 1\}^T$ and $\mathbf{v} \in \{0, 1\}^K$ are *actions* based on $\mathbf{m}$ and $\mathbf{n}$, and $\mathbf{u}_t = 1$ indicates the t-th frame in $\mathcal{V}$ is used; similarly $\mathbf{v}_k = 1$ means the k-th 3D convolution stage in the 3D model is activated. Zero entries in $\mathbf{u}$ and $\mathbf{v}$ represent inactive frames and convolutions, respectively. During training, $\mathbf{u}$ and $\mathbf{v}$ are produced by sampling from the corresponding policy, and a greedy approach is used at test time.

Given these actions, a subset $\mathcal{V}'$ of the full clip $\mathcal{V}$ is formed based on $\mathbf{u}$. Similarly, according to $\mathbf{v}$, certain 3D convolution layers are changed to 2D by taking only the center channel of its 3D convolutional filter along the temporal axis, *i.e.*, the slicing operation $k_{t\times d\times d}[\lfloor \frac{t}{2} \rfloor, :, :]$ in PyTorch style. Then, conditioned on $\mathcal{V}'$, we run a forward pass with the 3D network where certain 3D convolutions are degraded, and a prediction is then computed. To encourage correct predictions with limited computation, we evaluate these actions with a reward function:

$$R(\mathbf{x}) = \begin{cases} 1 - \mathcal{O}(\mathbf{x}) & \text{for correct prediction} \\ -\gamma & \text{else} \end{cases} \tag{4}$$

where $\mathcal{O}(\mathbf{x})$ represents the *normalized* computational cost of the action and $\mathbf{x} \in \{\mathbf{u}, \mathbf{v}\}$. Based on Eqn. 4, we compute two rewards for frame actions and convolution actions respectively, encouraging using as little computation as possible when making correct predictions while penalizing incorrect predictions with a negative reward, *i.e.*, $-\gamma$. Note that γ also balances the speed-accuracy trade-off with different values. While we instantiate $\mathcal{O}(\mathbf{u})$ and $\mathcal{O}(\mathbf{v})$ as $(\frac{||\mathbf{u}||_0}{T})$ and $(\frac{||\mathbf{v}||_0}{K})^2$—the normalized usage of the number of frames and 3D convolutions—there are also other options such as FLOPs [26, 15]. The selection network is then optimized to maximize the expected reward:

$$\max_{\mathbf{w}} \mathcal{L} = \mathbb{E}_{\mathbf{u}\sim\pi_f, \mathbf{v}\sim\pi_c}[R(\mathbf{u}) + R(\mathbf{v})]. \tag{5}$$

We use policy gradient methods [32] to learn the parameters $\mathbf{w}$ for the selection network and the expected gradient can be derived as:

$$\begin{aligned} \nabla_{\mathbf{w}}\mathcal{L} = \mathbb{E}\,[&R(\mathbf{u})\nabla_{\mathbf{w}}\log \pi^f(\mathbf{u} \,|\, \mathcal{V}) \\ &+ R(\mathbf{v})\nabla_{\mathbf{w}}\log \pi^c(\mathbf{v} \,|\, \mathcal{V})]. \end{aligned} \tag{6}$$

Eqn. 6 can be estimated with many samples at a time, and thus we use samples in mini-batches to compute the expected gradient and then Eqn. 6 is approximated by:

$$\begin{aligned} \nabla_{\mathbf{w}}\mathcal{L} \approx \frac{1}{B}\sum_{i=1}^{B} [&R(\mathbf{u}_i)\nabla_{\mathbf{w}}\log \pi^f(\mathbf{u}_i \,|\, \mathcal{V}_i) \\ &+ R(\mathbf{v}_i)\nabla_{\mathbf{w}}\log \pi^c(\mathbf{v}_i \,|\, \mathcal{V}_i)], \end{aligned} \tag{7}$$

where B is the total number of samples in the mini-batch. The gradient is then propagated back to train the policy network with SGD. We further reduce variance by adding a baseline function to the reward [32].

So far we have only trained the selection network while keeping the pretrained video model fixed. The selection network is able to learn decent policies that use fewer frames and 3D convolutions while maintaining prediction accuracies. However, input distributions to the 3D model are no longer the same as those used to train the original network, where all frames and 3D convolutions are used. As a result, the 3D model is not equipped with the ability to deal with inputs with varying number of frames and 3D convolutions that are adaptively turned on/off. To remedy this, we further jointly fine-tune the 3D model with the selection network such that it is able to accustomed to such adaptive inference paradigm. The objective function then becomes:

$$\min_{\mathbf{w},\boldsymbol{\theta}} -\sum_{j=1} \mathbf{y}^j \log(\mathbf{F}(\mathcal{V}; \boldsymbol{\theta})^j) - \mathcal{L}(\mathbf{w}) \tag{8}$$

[2] We consider turning off an entire 3D convolution stage that contains multiple 3D convolutional layers to save more computation.

Algorithm 1: Training algorithm of our approach.

```
   Input: An input video clip $\mathcal{V}$, the number of epochs
          of for training the selection network $E_1$, the
          number of epochs of joint fine-tuning $E_2$
1  Obtain a pretrained video classifier $\mathbf{F}$
2  Randomly initialize selection network $\mathbf{w}$
3  for $e \leftarrow 0$ to $E_1$ do
4      $\mathbf{m}, \mathbf{n} = \mathtt{sigmoid}(f_p(\mathcal{V}; \mathbf{w}))$
5      $\mathbf{u}, \mathbf{v} \sim \pi_{\mathbf{w}}(\mathbf{u}|\mathcal{V}), \pi_{\mathbf{w}}(\mathbf{v}|\mathcal{V})$          // Eqn. 3
6      $p = \mathbf{F}(\mathcal{V}|\mathbf{u}, \mathbf{v})$    // Apply actions on F and forward
7      $R = R(\mathbf{u}) + R(\mathbf{v})$                  // Eqn. 4
8      $\mathbf{w} = \mathbf{w} - \nabla \mathbf{w} \mathcal{L}$                     // Eqn. 6
9  end
10 for $e \leftarrow 0$ to $E_2$ do
11     Repeat Line 4-7
12     $\mathbf{w} = \mathbf{w} - \nabla \mathbf{w} \mathcal{L}$                     // Eqn. 6
13     $\boldsymbol{\theta} = \mathbf{w} - \nabla \boldsymbol{\theta} \mathcal{L}_{cls}$                  // Eqn. 8
14 end
```

where $\boldsymbol{\theta}$ denotes the weights of the 3D network $\mathbf{F}$ and the first term is the cross-entropy loss for an input clip $\mathcal{V}$ with one-hot label $\mathbf{y}$ for classification training. Algorithm 1 summarizes algorithm of Ada3D.

4. Experiments

4.1. Experimental Setup

Datasets and evaluation metrics. We evaluate our approach on three video recognition datasets: ActivityNet (ACTIVITYNET) [16], Fudan-Columbia Video Datasets (FCVID) [19] and Mini-Kinetics-200 (MINI-KINETICS) [44]. ACTIVITYNET contains around $20K$ Youtube videos of 200 action classes, with an average duration of 117 seconds. We use the latest version 1.3 and its official split with $10,024$ training videos, $4,926$ validation videos and $5,044$ testing videos. We report results on the validation set as the labels of testing videos are not publicly available. FCVID consists of $91,223$ Youtube videos belonging to 239 categories, with an average duration of 167 seconds. The official split is adopted with a training set of $45,611$ videos and a testing set of $45,612$ videos. MINI-KINETICS is a publicly released subset of KINETICS [4] initially introduced in [44], consisting of 200 classes with the most training samples in Kinetics; 400 and 25 videos are sampled from each action class for training and validation, forming a training set with $80,000$ videos and a validation set with $5,000$ videos. Here we use the identical samples as [44]. To demonstrate the transferability of the selection network, we experiment with the Kinetics full set, which contains 240K training videos and 20K validation videos.

Following official instructions, we report mean average precision (mAP) on ACTIVITYNET and FCVID. For MINI-KINETICS and KINETICS, we report Top-1 accuracy.

Network architectures. We use an I3D [4] with a backbone of ResNet-50 [14] as the 3D video model if not mentioned otherwise, due to its popularity and competitive recognition performance across various benchmarks [4, 13, 16]. Our implementation follows [7], where 3D convolutions are factorized spatially and temporally in a similar way as R(2+1)-D [35], which is already a more efficient architecture than original I3D. In addition, we also experiment with the Slowonly model introduced in [9] to demonstrate the compatibility of our approach with more recent networks.

We use a lightweight architecture for the selection network with negligible computational overhead. Specifically, we use MobileNetV2 [30] as the backbone of the selection network. The inputs to the network are downsampled to 112×112 per frame, and it only requires 0.08 GFLOPs to compute features for each frame.

Implementation details. All 3D networks are fine-tuned from models provided by [7], which are pre-trained on Kinetics. We fine-tune 3D models for 40 epochs on FCVID and ACTIVITYNET and 20 epochs on MINI-KINETICS, with a cosine learning rate schedule starting at 0.01 and a batch size of 64. The MobileNetV2 backbone of the selection network is also pre-trained on these datasets with the same schedules to speed up convergence. We first fix the pretrained 3D models and train the selection network for 40 epochs with a learning rate of 0.0001 and a batch size of 256. Finally, the whole pipeline is jointly fine-tuned for 60 epochs with the same learning rate described above. SGD with momentum 0.9 is used for optimization. We use 8 GPUs for all experiments.

Regarding network inputs during training, we follow [9, 8] by randomly sampling a clip with 8/16 frames using a temporal stride of 8 (sampling rate) from a given video. For the spatial domain, 224×224 pixels are randomly cropped from the sampled clip during training. For inference, we follow the common practice [9, 8, 39] and uniformly sample 10 clips with a spatial size of 256×256 from a testing video. Video-level prediction is obtained by averaging the clip-level predictions.

4.2. Main Results

We compare our proposed method with various baselines under different input settings (8/16 frames per clip) and report results in Table 1. The baselines we use include:

- *Random*: Based on the frame usage and convolution usage produced by Ada3D, we generate random policies that use a similar amount of computational resources compared to Ada3D.
- *Random FT*: The 3D model is further jointly fine-tuned with the random policies.

	FCVID				ActivityNet				Mini-Kinetics			
	mAP	GFLOPs	# 3D	# Frame	mAP	GFLOPs	# 3D	# Frame	Acc	GFLOPs	# 3D	# Frame
8-frame per clip												
Upper	82.1	58.6	5.0	8.0	82.6	58.6	5.0	8.0	79.0	58.6	5.0	8.0
Random	78.1	36.1	2.2	5.8	79.2	42.9	3.0	6.6	74.0	42.2	2.0	6.9
Random FT	80.7	36.1	2.2	5.8	81.1	42.9	3.0	6.6	77.4	41.5	1.9	6.8
Ours	81.9	35.6	2.2	5.7	82.6	42.2	3.1	6.6	78.9	42.4	1.9	6.9
16-frame per clip												
Upper	84.4	117.3	5.0	16.0	84.4	117.3	5.0	16.0	79.6	117.3	5.0	16.0
Random	79.2	63.2	2.1	10.3	80.4	73.3	3.0	11.2	75.2	75.8	2.9	11.8
Random FT	82.0	65.3	2.1	10.6	82.8	71.3	3.0	11.1	78.2	78.0	2.9	12.0
Ours	84.3	66.6	2.1	10.7	84.0	70.1	3.0	11.1	79.2	73.8	2.9	11.8

Table 1: **Recognition performance and computational cost of our method *vs*. baselines.** Two input settings are experimented, *i.e*. 8-frame setting (**Top**) and 16-frame setting (**Bottom**). # 3D and # Frame denote the number of 3D convolutions and frames usage per input clip respectively, averaged over the entire test set. See texts for more details.

- *Upper*: The original pretrained 3D model with all 3D convolutions and all frames used, which can be viewed as a performance "upperbound" of our method.

As shown in Table 1, under the 8-frame input setting, Ada3D obtains an mAP (accuracy for Mini-Kinetics) of 81.9%, 82.6% and 78.9%, requiring an average of 35.6, 42.2 and 42.4 GFLOPs per clip on FCVID, ActivityNet, Mini-Kinetics respectively. Ada3D achieves comparable recognition performance but brings 40%, 28% and 27% computational savings. This confirms that Ada3D is able to learn effective 3D convolution and frame usage policies by saving computational resources and preserving accuracies at the same time across different datasets. Similar patterns are also observed under the 16-frame input setting for all three datasets.

Using similar computational resources, Ada3D improves the *Random* baseline by 3.5% to 5% mAP/accuracy on three datasets. Ada3D also outperforms *Random FT* by 1% to 2.5%. These results verify that Ada3D produces adaptive polices and allocates computational resources on a per-input basis to maintain recognition performance. It is worth noting that there are slightly differences in computational savings on different datasets. This results from the fact that video categories in these datasets are different. For example, FCVID contains some classes of static objects and scenes like "bridge" and "temple", and thus we observe more computational savings than ActivityNet and Mini-Kinetics, which are more activity-focused; on Mini-Kinetics, where categories are motion-intensive, more computational resources are needed compared to FCVID and ActivityNet.

Recognition with varying computational budgets. As discussed in Section 3.2, the choice of γ in Eqn. 4 adjusts the amount of penalty on policies that produce incorrect predictions, and thus it controls the speed/accuracy trade-off. Here we report recognition accuracies of Ada3D under different computational budgets. As demonstrated in Fig. 3, our method is able to cover a wide range of speed/accuracy trade-offs and consistently outperforms *Random FT* with different computational budgets. For example, on ActivityNet, Ada3D obtains an mAP of 82.6%, 81.9% and 80.9% with an average of 42.2, 30.1 and 24.4 GFLOPs per clip respectively, while *Random FT* obtains 81.1%, 80.7% and 79.8% with 42.9, 32.6 and 25.4 GFLOPS per clip on average. Same patterns are also observed on FCVID.

Extension to clip selection. As mentioned in Sec. 2, our method is orthogonal and thus could be complementary to the line of clip selection methods [41, 43, 20, 12] for efficient video recognition. We validate our hypothesis by combining our method with AdaFrame [43]. Specifically, we use Ada3D as the backbone of AdaFrame to dynamically allocate computational resources conditioned on each input clip, as opposed to the original AdaFrame that uses the

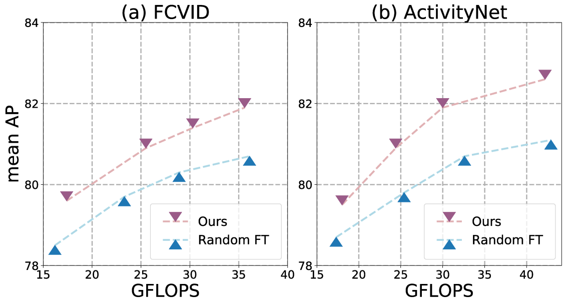

Figure 3: Recognition performance under different computational budgets controlled by γ.

	Van → *Ada*	*Van* → *Ada*	*Van* → *Ada*
# Clip	3.0 → 2.9	5.0 → 4.2	10.0 → 7.4
mAP	77.8 → 78.1	80.3 → 80.5	81.9 → 82.0

Table 2: **Extension to clip-level selection.** Combining Ada3D with AdaFrame [43] offers computational savings for video-level aggregation. # Clip denotes number of clips used per testing video; *Van* (Vanilla) and *Ada* denote our method without and with AdaFrame, respectively.

same amount of computation with a *fixed* backbone for all clips. Following [43], we train three variants of AdaFrame which operates on 3, 5, and 10 clips for different computational budgets. As demonstrated in Table 2, extending our approach with adaptive clip selection further decreases the computational cost while producing comparable performance with the *Upper*. For example, it reduces the number of clips sampled from each testing video from 10 to 7.4 and obtains an mAP of 82.0% that is on par with *Upper* (82.1%). Additionally, we believe our method is also complementary to other clip selection methods leveraging multi-modal inputs such as audio [20, 12], as well as adaptive spatial resolution modulating methods [26, 36, 46].

Method	Acc	GFLOPs	# 3D	# Frame
Upper	73.1	58.6	5.0	8.0
Ours	72.8	43.7	2.3	6.9

Table 3: **Transferring learned policies.** We fine-tune a Kinetics pretrained model on Kinetics full training set, with policies learned on Mini-Kinetics, and evaluate on Kinetics validation set.

Transferring learned policies. We now analyze whether the policies learned by our method can be transferred to novel action categories. To this end, we take the selection network trained on MINI-KINETICS and fine-tune a pretrained I3D model with a ResNet-50 as its backbone on full Kinetics. We keep the weights of the selection network fixed during fine-tuning. Details of training and testing are the same as joint fine-tuning as described in Sec. 4.1. As shown in Table 3, policies learned on MINI-KINETICS can reduce the overall computational cost of the fine-tuned video model by 25% on Kinetics with negligible difference in recognition accuracy compared to the *Upper* baseline, indicating that our method learns strategies that are transferable to unseen classes and videos. It is worth noting that the I3D baseline we use obtains superior recognition performance on Kinetics that is higher than [4, 44] and competitive compared to results reported in [39] using 32 frames per input clip.

Compatibility with different 3D architectures. Next, we evaluate the compatibility of our approach with different 3D networks. We use a more efficient 3D network architecture recently introduced in [9] termed as Slowonly and evaluate our approach. In particular, it only uses 3D convolutions in the 4-th and 5-th stage of a ResNet50, resulting in competitive recognition performance with less computational cost. As shown in Table 4, our method still obtains 20% to 40% savings in GFLOPs with similar recognition performance, indicating Ada3D is compatible with different 3D models. Our method by design is model-agnostic, for which we believe it could be complementary to recent work on designing efficient 3D models such as X3D [8] as well.

Method	mAP	GFLOPs	# 3D	# Frame
Upper	82.6	54.5	2.0	8.0
Ours	82.4	42.1	1.3	6.6
Upper	83.5	109.1	2.0	16.0
Ours	83.4	61.8	1.4	9.5

Table 4: Results on FCVID [19] using Slowonly [9] architecture as 3D model. **Top**: 8-frame input setting. **Bottom**: 16-frame input setting.

Qualitative analysis. In addition to the quantitative results presented above, we also qualitatively analyze our method. In particular, we observe that our method produces policies with fewer 3D convolutions and frames for input clips that are more "static", while uses more for motion-intensive instances. As shown in Fig. 4, a smaller number of 3D convolutions and frames are applied on clips with discriminative static cue. For instance, the presence of "bass" and "book binder" for class "playing bass guitar" and "book binding" suffice to produce correct predictions, and the scene of a "court" serves as a strong contextual signal for "hurling". On the other hand, for motion-intensive action classes and instances, especially those related to human movement such as "breakdancing", "somersaulting" and "Tai Chi", more computational resources are allocated by our method to capture finer temporal relationships among frames.

4.3. Discussion

Impact of joint finetuning. Recall that we first train the seletion network with the 3D model fixed and then jointly fine-tune both of them. Here we analyze the performance of our method without the first selection network training stage (Tr) or the joint fine-tuning stage (FT). For faster evaluation, we uniformly sample 3 clips from each test video. Results are shown in Table 5.

As can be seen, joint fine-tuning is crucial to further improve the recognition performance (75.9 *vs*. 77.8). This indicates that fine-tuning the video model together with learned policies indeed helps the 3D model to adapt to the adaptive inference paradigm brought by the selection network. It is worth noting that skipping the first training

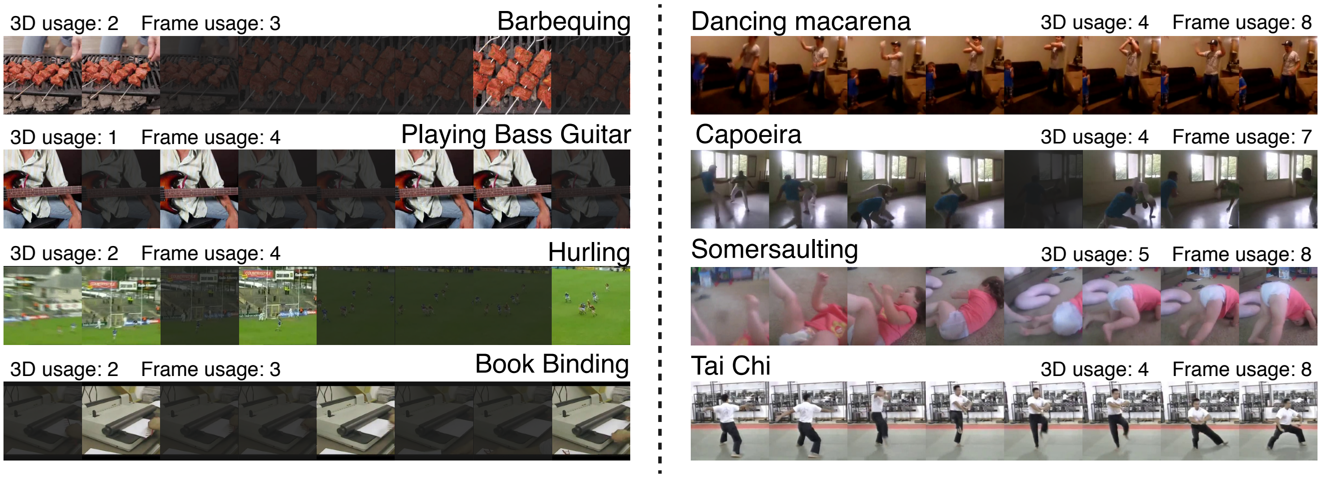
3D usage: 2 Frame usage: 3 Barbequing
3D usage: 1 Frame usage: 4 Playing Bass Guitar
3D usage: 2 Frame usage: 4 Hurling
3D usage: 2 Frame usage: 3 Book Binding
Dancing macarena 3D usage: 4 Frame usage: 8
Capoeira 3D usage: 4 Frame usage: 7
Somersaulting 3D usage: 5 Frame usage: 8
Tai Chi 3D usage: 4 Frame usage: 8

Figure 4: **Qualitative results.** Black mask indicates the frame is discarded. **Left:** Fewer 3D convolutions and frames are used for action classes and instances that are more "static", *i.e.* containing discriminative static cue and contextual information. **Right:** For motion-intensive instances, more computation is allocated for probing finer temporal information.

		FCVID		ACTIVITYNET	
Tr	FT	mAP	GFLOPs	mAP	GFLOPs
Upper		78.1	58.6	76.4	58.6
		72.3	36.1	71.1	42.9
✓		75.9	34.7	74.3	38.1
	✓	76.5	34.9	75.1	37.6
✓	✓	77.8	35.6	76.1	42.2

Table 5: Ablation on the effectiveness of two training stages.

stage (*i.e.*, directly training the selection network with the 3D model jointly) leads to a lower recognition performance (76.5 *vs*. 77.8). We posit the reason is that adding another objective (the classification loss) while training the selection network from random initialization further increases the instability of network learning under such a reinforcement learning setting; and thus the selection network converges to sub-optimal policies.

		FCVID		ACTIVITYNET	
3D	Frame	mAP	GFLOPs	mAP	GFLOPs
Upper		78.1	58.6	76.4	58.6
		75.5	35.6	74.3	42.3
✓		76.8	35.3	75.3	41.1
	✓	76.3	35.5	74.8	43.5
✓	✓	77.8	35.6	76.1	42.2

Table 6: Ablation on the usefulness of 3D convolution usage and frame usage policies.

Contributions of convolution and frame usage policies. To demonstrate the effectiveness of 3D convolution usage and frame usage policies learned by the two-head selection network, we conduct experiments to analyze contributions of the two components. In particular, we replace each/both components with randomly generated policies similar to *Random FT*. Here we use 3-clip testing as well. As shown in Table 6, applying either 3D or frame usage policy improves recognition performance under the same computational budget, while using both achieves the best performance with 1% improvement over the single-component settings, indicating the double-head architecture can learn to produce policies cooperatively.

5. Conclusion

We presented Ada3D, a framework that learns to derive adaptive 3D convolution and frame usage policies—determining which 3D convolutions in a pretrained 3D video model and which frames in the input clip to use on a per-input basis—for efficient video recognition. In particular, a two-head selection network is trained with policy gradient methods to produce these policies, reducing overall computational cost while maintaining recognition performance. Extensive experimental results on three large-scale video recognition datasets indicate that Ada3D achieves 20%-50% computational savings on state-of-the-art 3D video models while achieving similar accuracies. We further demonstrate Ada3D is compatible with different backbones of 3D model and other clip selection methods, and qualitatively show that more computational resource is allocated on motion-intensive instances but less on static ones by Ada3D.

Acknowledgement This work is supported by IARPA via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00345.

References

[1] The state of online video for 2020. https://www.forbes.com/sites/tjmccue/2020/02/05/looking-deep-into-the-state-of-online-video-for-2020/. 1

[2] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. In *ICLR*, 2020. 3

[3] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. In *ICML*, 2017. 3

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 3, 5, 7

[5] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *ECCV*, 2018. 2

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[7] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. Pyslowfast. https://github.com/facebookresearch/slowfast, 2020. 5

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

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

[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[11] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *CVPR*, 2017. 3

[12] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. In *CVPR*, 2020. 2, 3, 6, 7

[13] 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*, 2017. 1, 2, 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5

[15] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, 2018. 4

[16] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 1, 2, 5

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[18] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 3

[19] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE TPAMI*, 2018. 2, 5, 7

[20] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019. 2, 3, 6, 7

[21] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2008. 4

[22] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *ICCV*, 2019. 3

[23] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. Videolstm convolves, attends and flows for action recognition. *CVIU*, 2018. 2

[24] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 2

[25] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NIPS*, 2017. 3

[26] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *ECCV*, 2020. 3, 4, 7

[27] Mahyar Najibi, Bharat Singh, and Larry S Davis. Autofocus: Efficient multi-scale inference. In *ICCV*, 2019. 3

[28] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 4

[29] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 2

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 2, 5

[31] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2

[32] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998. 2, 4

[33] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: Generic features for video analysis. In *ICCV*, 2015. 1, 2, 3

[34] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019. 1, 2

[35] 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. 1, 2, 3, 5

[36] Burak Uzkent and Stefano Ermon. Learning when and where to zoom with deep reinforcement learning. In *CVPR*, 2020. 3, 7

[37] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 3

[38] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE TPAMI*. 2

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 5, 7

[40] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 3

[41] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, and Shilei Wen. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *ICCV*, 2019. 2, 3, 6

[42] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018. 3

[43] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *CVPR*, 2019. 2, 3, 6, 7

[44] 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, 3, 5, 7

[45] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *CVPR*, 2020. 1, 3

[46] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *CVPR*, 2020. 3, 7

[47] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[48] Yin-Dong Zheng, Zhaoyang Liu, Tong Lu, and Limin Wang. Dynamic sampling networks for efficient action recognition in videos. *IEEE TIP*, 2020. 2

[49] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 2

[50] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *ECCV*, 2018. 2