Dynamic Neural Network for Efficient Video Recognition

By

Bowen Pan

B.S., Shanghai Jiao Tong University (2019)

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2023

Authored by: Bowen Pan
Department of Electrical Engineering and Computer Science
May 17, 2023

Certified by: Aude Oliva
Senior Research Scientist of the Computer Science and Artificial Intelligence Laboratory
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Dynamic Neural Network for Efficient Video Recognition

by

Bowen Pan

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Recognizing real-world videos is a challenging task that requires the use of deep learning models. These models, however, require extensive computational resources to achieve robust recognition. One of the main challenges when dealing with real-world videos is the high correlation of information across frames. This results in redundancy in either temporal or spatial feature maps of the models, or both. The amount of redundancy largely depends on the dynamics and events captured in the video. For example, static videos typically have more temporal redundancy, while videos focusing on objects tend to have more channel redundancy.

To address this challenge, we propose a novel approach that reduces redundancy by using an input-dependent policy to determine the necessary features for both temporal and channel dimensions. By doing so, we can identify the most relevant information for each frame, thus reducing the overall computational load. After computing the necessary features, we reconstruct the remaining redundant features from those using cheap linear operations. This not only reduces the computational cost of the model but also keeps the capacity of the original model intact.

Moreover, our proposed approach has the potential to improve the accuracy of real-world video recognition by reducing overfitting caused by the redundancy of information across frames. By focusing on the most relevant information, our model can better capture the unique characteristics of each video, resulting in more accurate predictions. Overall, our approach represents a significant step forward in the field of real-world video recognition and has the potential to enable the development of more efficient and accurate deep learning models for this task.

Thesis Supervisor: Aude Oliva
Title: Senior Research Scientist

Acknowledgments

I would like to express my deepest gratitude to my family for their unwavering support and encouragement throughout my academic journey. Their love, patience, and understanding have been invaluable to me.

I am also immensely grateful to my advisor, Aude Oliva, for her guidance, wisdom, and expertise. Her mentorship has been instrumental in shaping my research and helping me grow as a researcher.

I would like to acknowledge MIT-IBM Watson AI Lab for generously sponsoring my research and providing me with access to their resources. Their support has been instrumental in enabling me to pursue my research goals.

I would like to thank my collaborator Ramesware Panda, as well as Rogerio Feris, Camilo Fosco, and Alex Andonian, for their invaluable contributions to my research. Their insights, feedback, and collaboration have been critical to the success of my work.

I am also grateful to my lab mates for creating a supportive and stimulating research environment. Their friendship, encouragement, and constructive feedback have been invaluable to me.

Finally, I would like to express my gratitude to my friends for their unwavering support, encouragement, and understanding. Their love and laughter have been a constant source of joy and inspiration.

Thank you all for your invaluable contributions to my academic journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Large, computationally expensive models based on 2D or 3D convolutional neural networks (CNNs) have become the cornerstone of video understanding in recent years, as evidenced by numerous studies [44, 4, 46]. As a result, the pursuit of increasing computational efficiency has emerged as a critical area of interest and research [9, 62, 63]. Despite the fact that the majority of these efficiency-driven approaches primarily concentrate on architectural modifications in order to maximize network capacity while maintaining a compact model [63, 9], or on refining the network's ability to effectively process temporal information [10, 28], they often neglect to address the unnecessary computations performed by CNNs at various levels of the network [16, 21, 40, 9, 37].

This oversight is particularly relevant for video models, given the high appearance similarity between consecutive frames that inevitably leads to a considerable amount of redundancy. This redundancy not only increases computational demands but also slows down the processing time for video recognition tasks, making it a critical issue to address in order to improve the overall efficiency and effectiveness of video understanding models.

In light of this, it becomes increasingly important for researchers and practitioners to develop novel techniques and approaches that can systematically identify and eliminate these redundancies while preserving the network's ability to accurately

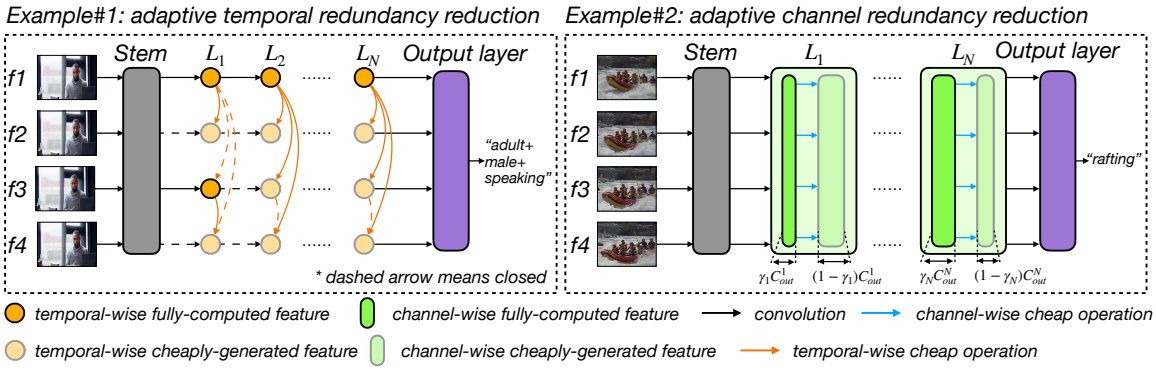


Figure 1-1: Our VA-RED^2 framework dynamically reduces the redundancy in two dimensions. Example 1 (left) shows a case where the input video has little movement. The features in the temporal dimension are highly redundant, so our framework fully computes a subset of features, and reconstructs the rest with cheap linear operations. In the second example, we show that our framework can reduce computational complexity by performing a similar operation over channels: only part of the features along the channel dimension are computed, and cheap operations are used to generate the rest.

interpret and understand video data. By doing so, the field of video recognition can make significant strides in optimizing computational resources and reducing the processing time required for a wide range of applications, from video analytics and content-based retrieval to real-time surveillance and automated video editing.

1.2 Motivation and Goal

In this paper, our primary objective is to dynamically reduce the internal computations of widely-used video CNN architectures by effectively tackling the internal redundancy that pervades both time and channel dimensions in video models. Our motivation stems from the realization that feature maps exhibit a high degree of similarity in these dimensions, with the level of redundancy varying depending on the specific input. For instance, static videos are characterized by a higher degree of temporal redundancy, whereas videos that depict a single large object in motion tend to generate a greater number of redundant feature maps.

To address and combat this varied redundancy across channel and temporal dimensions, we introduce an innovative, input-dependent redundancy reduction framework specifically tailored for efficient video recognition (refer to Figure 1-1 for a visual

representation). This cutting-edge framework dynamically adjusts its operations based on the input, allowing it to effectively minimize redundancy while maintaining high-quality recognition results.

A significant advantage of our approach is its model-agnostic nature, which means it can be seamlessly integrated into any state-of-the-art video recognition networks without the need for extensive modifications or adaptations. This versatility ensures that our framework has the potential to greatly impact the field of video recognition, improving efficiency and reducing computational demands across a wide range of existing and future network architectures.

By dynamically reducing internal computations and addressing redundancy in a targeted manner, our framework paves the way for more efficient and effective video recognition tasks. This, in turn, has the potential to revolutionize various applications of video recognition, from surveillance and security systems to video content analysis and beyond.

1.3 Our Idea

Our framework significantly enhances efficiency by strategically replacing full computations of certain redundant feature maps with more cost-effective reconstruction operations. To accomplish this, our framework deliberately avoids computing all feature maps. Instead, it focuses on calculating only the non-redundant portions of feature maps and subsequently reconstructing the remaining maps using efficient linear operations derived from the non-redundant feature maps. This targeted approach helps to streamline the computation process.

Moreover, our framework is designed to adapt its decision-making process on a per-input basis, ensuring optimal efficiency for each unique input. It achieves this by learning an input-dependent policy that establishes a "full computation ratio" for every layer of a 2D/3D network. This crucial ratio determines the proportion of features that will be fully computed at a given layer, as opposed to the features that will be reconstructed using the non-redundant feature maps as a basis.

It is important to note that this strategic approach is applied consistently across both time and channel dimensions, further optimizing the efficiency of the framework. Through our extensive testing, we demonstrate that this innovative method significantly reduces the total floating-point operations (FLOPs) on a variety of common video datasets, all without sacrificing the accuracy of the results. This holds true when our method is applied to traditional video models, such as I3D [4] and R(2+1)D [46], as well as more advanced models like X3D [9]. The success of our approach in maintaining accuracy while reducing computational requirements showcases the potential of our framework in revolutionizing video recognition tasks.

1.4 Contributions

The primary **contributions** of our work encompass several key aspects: (1) We present a groundbreaking **input-dependent adaptive framework** for efficient video recognition, which automatically determines the feature maps to compute for each input instance. This approach stands in stark contrast to the majority of existing video processing networks, where feature redundancy across both time and channel dimensions is not directly addressed or mitigated. (2) We propose an **adaptive policy** that is jointly learned with the network weights using a fully differentiable method and a shared-weight mechanism. This enables us to make informed decisions regarding the number of feature maps to compute at any given time. Our approach is model-agnostic, allowing it to be applied to any backbone network and effectively reduce feature redundancy in both time and channel domains. (3) We demonstrate the **impressive results of our framework when compared to baseline models**, achieving a 30% reduction in computation relative to R(2+1)D [46], a 40% reduction compared to I3D-InceptionV2 [4], and approximately 20% less computation than the recently proposed X3D-M [9], all without sacrificing performance in video action recognition tasks. We extensively test the superiority of our approach on three video recognition datasets (Mini-Kinetics-200, Kinetics-400 [4], and Moments-In-Time [36]) and one spatio-temporal action localization dataset (J-HMDB-21 [25]), showcasing its

effectiveness. (4) We also present a **generalization of our framework** to various tasks, such as video action recognition, spatio-temporal localization, and semantic segmentation. In doing so, we achieve promising results while delivering significant computational reductions compared to competing methods, further highlighting the potential of our proposed framework.

Chapter 2

Related Work

2.1 Efficiency in Video Understanding Models

In recent years, video understanding has made significant progress, thanks to the adoption of convolutional neural networks (CNNs), specifically 2D CNNs [26, 41, 5, 11, 13, 49, 59, 31, 8] or 3D CNNs [44, 4, 18, 46]. While these networks have shown promising results on common benchmarks, there is a growing interest in developing more efficient techniques and smaller models that can still deliver reasonable performance.

To achieve this goal, previous works have explored various approaches such as hybrid 2D-3D architectures [53, 62, 63], group convolution [45], and selecting salient clips [28]. Other approaches have attempted to reduce the amount of temporal information consumed by the network, such as Feichtenhofer et al.'s dedicated low-framerate pathway [10].

Moreover, researchers have proposed expansion of 2D architectures through a stepwise expansion approach over key variables such as temporal duration, frame rate, spatial resolution, and network width, as recently proposed in [9]. Additionally, some works have focused on learning motion dynamics of videos with a self-supervised task for video understanding [6], incorporating an efficient learnable 3D-shift module into a 3D video network [7], devising a correlation module to learn correlation along the temporal dimension [48], and encoding the clip-level ordered temporal information with a CIDC network [30].

While these approaches have brought considerable efficiency improvements, none of them dynamically calibrates the required feature map computations on a per-input basis. In contrast, our proposed framework achieves substantial improvements in average efficiency by avoiding redundant feature map computation depending on the input. This approach has the potential to enable the development of even more efficient and accurate video understanding models in the future.

2.2 Adaptive Inference

In recent years, there has been a surge in the development of adaptive computation methods aimed at improving efficiency in deep learning models [1, 2, 47, 50, 14, 35]. These methods typically add decision branches to different layers of CNNs, enabling the network to learn whether to exit the network for faster inference. For instance, Yu et al. [57] proposed adding decision branches to each layer of a network to enable dynamic network width, while Wang et al. [50] proposed skipping convolutional blocks on a per-input basis using reinforcement learning and supervised pre-training. Veit et al. [47] proposed a block-skipping method controlled by samples from a Gumbel softmax, while Wu et al. [51] developed a reinforcement learning approach to achieve this goal.

Adaptive computation time for recurrent neural networks is also presented in [14]. SpotTune [15] learns to route information through finetuned or pre-trained layers adaptively. Several recent works have focused on selecting salient frames conditioned on the input [55, 52, 28, 12] while recognizing actions in long untrimmed videos.

In contrast, our goal in this paper is to remove feature map redundancy by dynamically calibrating the necessary computations for temporal and channel dimensions on a per-input basis. Our approach is different from adaptive data sampling [55, 52, 28, 12], which focuses on selecting salient frames. Our method reduces redundancy in both temporal and channel dimensions, making it applicable to both 3D and 2D models.

Moreover, our method integrates all the inference routes into a single model, which is almost the same size as the original base model. This is in contrast to AR-Net [34],

which recently learned to adaptively choose the resolution of input frames with several individual backbone networks for video inference. However, AR-Net is only applicable to 2D models and is focused on spatial resolution, while our method is applicable to both 2D and 3D models and focuses on reducing redundancy in both temporal and channel dimensions. Additionally, our method is significantly smaller than AR-Net in terms of the number of model parameters.

Overall, the proposed approach has the potential to enable the development of even more efficient and accurate video understanding models in the future. It represents a significant step forward in the field of adaptive computation and has practical implications for a wide range of video recognition tasks.

2.3 Neural Architecture Search

Automated architecture search has been a popular research direction in recent years, with several approaches proposed for discovering optimal architectures for deep learning models. Liu et al. [32] have formulated the architecture search task in a differentiable manner, enabling efficient optimization through gradient descent. Cai et al. [3] have proposed a method for directly learning architectures for a target task and hardware, while Tan et al. [43] have designed a compound scaling strategy that searches through several key dimensions for CNNs (depth, width, resolution). Furthermore, Tan et al. [42] have incorporated latency into the architecture search process to find efficient networks adapted for mobile use.

Our approach is different from these methods, as it learns a policy that chooses between full or reduced convolutions at inference time. By effectively switching between various discovered subnetworks, our method minimizes redundant computations while delivering high accuracy. This approach is unique in its focus on reducing redundancy in both temporal and channel dimensions, making it applicable to both 2D and 3D models.

In addition, our approach is inspired by recent work on adaptive computation, which has also focused on optimizing the computations required for deep learning

models. For instance, Yu et al. [57] proposed a method for dynamically adjusting the width of CNNs, while Wang et al. [50] proposed a method for skipping convolutional blocks on a per-input basis. These approaches are complementary to our method, as they aim to optimize different aspects of the computation required for deep learning models.

Overall, the proposed method represents a significant step forward in the field of automated architecture search and adaptive computation. It has practical implications for a wide range of video recognition tasks and has the potential to enable the development of even more efficient and accurate deep learning models in the future.

Chapter 3

Method

Our main goal is to automatically decide which feature maps to compute for each input video in order to classify it correctly with the minimum computation. The intuition behind our proposed method is that there are many similar feature maps along the temporal and channel dimensions. For each video instance, we estimate the ratio of feature maps that need to be fully computed along the temporal dimension and channel dimension. Then, for the other feature maps, we reconstruct them from those pre-computed feature maps using cheap linear operations.

3.1 Approach Overview

Without loss of generality, we start from a 3D convolutional network $\mathcal{G}$, and denote its l^{th} 3D convolution layer as f_l, and the corresponding input and output feature maps as X_l and Y_l respectively. For each 3D convolution layer, we use a very lightweight policy layer p_l denoted as *soft modulation gate* to decide the ratio of feature maps along the temporal and channel dimensions which need to be computed. As shown in Figure 3-1, for temporal-wise dynamic inference, we reduce the computation of 3D convolution layer by dynamically scaling the temporal stride of the 3D filter with a factor $R = 2^{p_l(X_l)[0]}$. Thus the shape of output Y_l' becomes $C_{out} \times T_o/R \times H_o \times W_o$.

To keep the same output shape, we reconstruct the remaining features based on Y_l' as

$$Y_l[j+iR] = \begin{cases} \Phi_{i,j}^t(Y_l'[i]) & \text{if } j \in \{1, ..., R-1\} \\ Y_l'[i] & \text{if } j = 0 \end{cases}, i \in \{0, 1, ..., T_o/R - 1\}, \tag{3.1}$$

where $Y_l[j+iR]$ represents the $(j+iR)^{th}$ feature map of Y_l along the temporal dimension, $Y_l'[i]$ denotes the i^{th} feature map of Y_l', and $\Phi_{i,j}^t$ is the cheap linear operation along the temporal dimension. The total computational cost of this process can be written as:

$$\mathcal{C}(f_l^t) = \frac{1}{R} \cdot \mathcal{C}(f_l) + \sum_{i,j} \mathcal{C}(\Phi_{i,j}^t) \approx \frac{1}{R} \cdot \mathcal{C}(f_l), \tag{3.2}$$

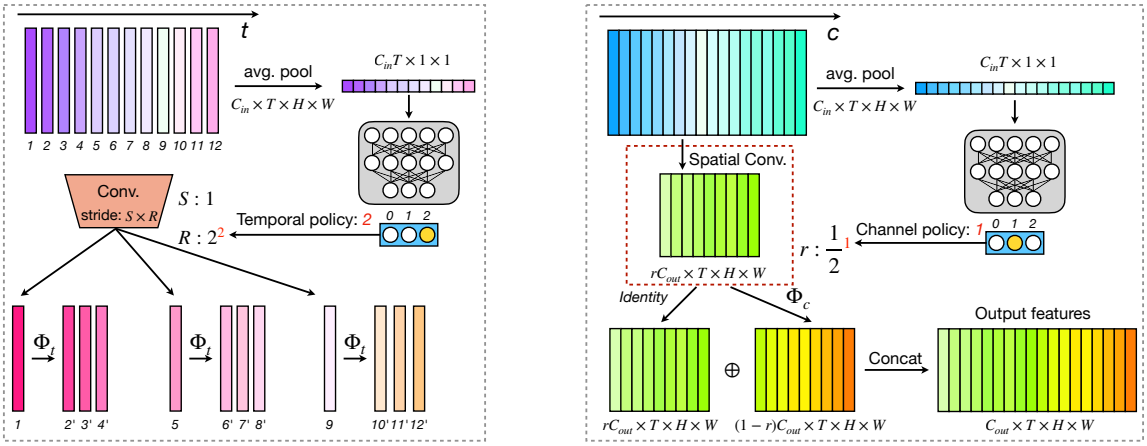
where the function $\mathcal{C}(\cdot)$ returns the computation cost for a specific operation, and f_l^t represents our dynamic convolution process along temporal dimension. Different from temporal-wise dynamic inference, we reduce the channel-wise computation by dynamically controlling the number of output channels. We scale the output channel number with a factor $r = (\frac{1}{2})^{p_l(X_l)[1]}$. In this case, the shape of output Y_l' is $rC_{out} \times T_o \times H_o \times W_o$. Same as before, we reconstruct the remaining features via cheap linear operations, which can be formulated as $Y_l = [Y_l', \Phi^c(Y_l')]$, where $\Phi^c(Y_l') \in R^{(1-r)C_{out} \times T_o \times H_o \times W_o}$ represents the cheaply generated feature maps along the channel dimension, and $Y_l \in R^{C_{out} \times T_o \times H_o \times W_o}$ is the output of the channel-wise dynamic inference. The total computation cost of joint temporal-wise and channel-wise dynamic inference is:

$$\mathcal{C}(f_l^{t,c}) \approx \frac{r}{R} \cdot \mathcal{C}(f_l), \tag{3.3}$$

where $f_l^{t,c}$ is the adjunct process of temporal-wise and channel-wise dynamic inference.

3.2 Soft Modulation Gate for Differentiable Optimization

We adopt an extremely lightweight policy layer p_l called soft modulation gate for each convolution layer f_l to modulate the ratio of features which need to be computed.



(*a*) *Temporal-wise dynamic convolution* (*b*) *Channel-wise dynamic convolution*

Figure 3-1: An illustration of dynamic convolution along temporal dimension (a) and channel dimension (b) respectively. Φ_t and Φ_s represent the temporal cheap operation and spatial cheap operation respectively. In (a), we multiply the temporal stride S with the factor $R = 2^{p_t}$ to reduce computation, where p_t is the temporal policy output by soft modulation gate. In (b), we compute part of output features with the ratio of $r = (\frac{1}{2})^{p_c}$, where p_c is the channel policy. Best viewed in color.

Specifically, the soft modulation gate takes the input feature maps X_l as input and learns two probability vectors $V_t^l \in R^{S_t}$ and $V_c^l \in R^{S_c}$, where S_t and S_c are the temporal search space size and the channel search space size respectively. The V_t^l and V_c^l are learned by:

$$[V_t^l, V_c^l] = p_l(X_l) = \phi(\mathcal{F}(\omega_{p,2}, \delta(\mathcal{N}(\mathcal{F}(\omega_{p,1}, G(X_l))))) + \beta_p^l), \tag{3.4}$$

where $\mathcal{F}(\cdot,\cdot)$ denotes the fully-connected layer, $\mathcal{N}$ is the batch normalization, $\delta(\cdot)$ represents the $\tanh(\cdot)$ function, G is the global pooling operation whose output shape is $C_{in} \cdot T \times 1 \times 1$, $\phi(\cdot)$ is the output activation function, here we just use $\max(\tanh(\cdot), 0)$ whose output range is $[0, 1)$, and $\omega_{p,1} \in R^{(S_t+S_c)\times D_h}$, $\omega_{p,2} \in R^{D_h \times C_{in} \cdot T}$ are the weights of their corresponding layers, D_h is the hidden dimension number. V_t^l and V_c^l will then be used to modulate the ratio of the feature maps to be computed in temporal-wise dynamic convolution and channel-wise dynamic convolution. During training, we obtain the final output of the dynamic convolution by weighted sum of all the feature

maps which contains different ratio of fully-computed features as follows:

$$Y_c^l = \sum_{i=1}^{S_c} V_c^l[i] \cdot f_l^c(X_l, r = (\frac{1}{2})^{(i-1)}), \quad Y_l = \sum_{j=1}^{S_t} V_t^l[j] \cdot f_l^t(Y_c^l, R = 2^{(j-1)}), \tag{3.5}$$

where $f_l^c(\cdot, r)$ is the channel-wise dynamic convolution with the channel scaling factor r, and $f_l^t(\cdot, R)$ it the temporal-wise dynamic convolution with the temporal stride scaling factor R. During the inference phase, only the dynamic convolutions whose weights are not zero will be computed.

3.3 Shared-weight Training and Inference

Many works in adaptive computation and neural architecture search suffer from very heavy computational cost and memory usage during training stage due to the large search space. In our case, under the naive implementation, the training computational cost and parameter size would linearly grow as the search space size increases. To train our model efficiently, we utilize a weight-sharing mechanism to reduce the computational cost and training memory. To be specific, we first compute all the possible necessary features using a big kernel. Then, for each dynamic convolution with different scaling factor, we sample its corresponding ratio of necessary features and reconstruct the rest features by cheap operations to get the final output. Though this, we are able to keep the computational cost at a constant value invariant to the search space. More details on this are included in Section A.2 of the Appendix.

3.4 Efficiency Loss

To encourage our network to output a computational efficient subgraph, we introduce the efficiency loss $\mathcal{L}_c$ during the training process, which can be formulated as

$$\mathcal{L}_e = (\mu_0 \sum_{l=1}^{L} \frac{\mathcal{C}(f_l)}{\sum_{k=1}^{L} \mathcal{C}(f_k)} \cdot \frac{r_l^s}{R_l^s})^2, \mu_0 = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}, \tag{3.6}$$

where r_l^s is channel scaling factor of the largest filter in the series of channel-wise dynamic convolutions, and R_l^s is stride scaling factor of the largest filter of temporal-wise dynamic convolutions. Overall, the loss function of our whole framework can be written as $\mathcal{L} = \mathcal{L}_a + \lambda_e \mathcal{L}_e$, where L_a is the accuracy loss of the whole network and λ_e is the weight of efficiency loss which can be used to balance the importance of the optimization of prediction accuracy and computational cost.

Chapter 4

Experiments

4.1 Datasets

We conduct our **video action recognition** experiments on three standard benchmarks: Mini-Kinetics-200, Kinetics-400, and Moments-In-Time. Mini-Kinetics-200 (assembled by [34]) is a subset of full Kinetics dataset [4] containing 121k videos for training and 10k videos for testing across 200 action classes. Moments-In-Time dataset has 802,244 videos in training and 33,900 videos in validation across 339 categories. To show the generalization ability to different task, we also conduct the **video spatio-temporal action localization** on J-HMDB-21 [25]. J-HMDB-21 is a subset of HMDB dataset [29] which has 928 short videos with 21 action categories. We report results on the first split. For **semantic segmentation** experiments, we use ADE20K dataset [60, 61], containing 20k images for training and 2k images for validation. ADE20K is a densely labeled image dataset where objects and object parts are segmented down to pixel level. We report results on validation set.

4.2 Model Architectures

We evaluate our method on three most widely-used model architectures: I3D [4], R(2+1)D [46], and the recent efficient model X3D [9]. We consider I3D-InceptionV2 (denoted as I3D below) and R(2+1)D-18 (denoted as R(2+1)D below) as our base

model. In our implementation of X3D, we remove all the swish non-linearity [39] except those in SE layer [22] to save training memory and speed up the inference speed on GPU. We choose X3D-M (denote as X3D below) as our base model and demonstrate that our method is generally effective across datasets.

4.3 Implementation Details

We train and evaluate our baseline models by mainly following the settings in their original papers [46, 53, 9]. We train all our base and dynamic models for 120 epochs on mini-Kinetics-200, Kinetics-400, and 60 epochs on Moments-In-Time dataset. We use a mini-batch size of 12 clips per GPU and adopt synchronized SGD with cosine learning rate decaying strategy [33] to train all our models. Dynamic models are finetuned with efficiency loss for 40/20 epochs to reduce density of inference graph while maintaining the accuracy. During finetuning, we set λ_c to 0.8 and learning rate to 0.01 for R(2+1)D and 0.1 for I3D and X3D. For testing, we adopt *K-LeftCenterRight* strategy: K temporal clips are uniformly sampled from the whole video, on which we sample the left, center and right crops along the longer spatial axis, the final prediction is obtained by averaging these $3 \times K$ clip predictions. We set $K = 10$ on Mini-Kinetics-200 and Kinetics-400 and $K = 3$ on Moments-In-Time. More implementation details are included in Section A.2 of Appendix. For video spatio-temporal action localization, we adopt YOWO architecture in [27] and replace 2D branch with 3D backbone to directly compare them. We freeze the parameters of 3D backbone as suggested in [27] due to small number of training video in J-HMDB-21 [25]. The rest part of the network is optimized by SGD with initial learning rate of 10^{-4}. Learning rate is reduced with a decaying factor of 0.5 at 10k, 20k, 30k and 40k iterations. For semantic segmentation, we conduct experiments using PSPNet [58], with dilated ResNet-18 [56, 20] as our backbone architecture. As PSPNet is devised for image semantic segmentation, we only apply the channel-wise redundancy reduction to the model and adopt synchronized SGD training for 100k iterations across 4GPUs with 2 images on each GPU. The learning rate decay follows the cosine learning rate schedule

Table 4.1: **Action recognition results using different number of input frames and different search space.** We choose R(2+1)D-18 on Mini-Kinetics-200 and study the performance with different number of input frames and different search space (denoted as Sea. Sp.). Search space of 2 means that both temporal-wise and channel-wise policy network have 2 alternatives: computing all feature maps, or computing only $\frac{1}{2}$) of the feature maps. Similarly, search space 3 have 3 alternatives: computing *1)* all feature maps, *2)* $\frac{1}{2}$ of feature maps, *3)* $\frac{1}{4}$ of feature maps. ✗ denote the base model and ✓ denote the dynamic model trained using our proposed approach VA-RED^2. We also report the average speed of different models in terms of number of clips processed in one second (*clip/second*).

length	sp.	GFLOPs$_{Avg}$	GFLOPs$_{Max}$	GFLOPs$_{Min}$	avg speed	clip-1	video-1
8	✗	27.7	27.7	27.7	192.1	56.4	66.8
	2	20.0(−28%)	22.1(−20%)	18.0(−35%)	**205.5**	57.7	**68.0**
	3	21.6(−22%)	23.2(−16%)	19.8(−29%)	201.4	**58.2**	67.7
16	✗	55.2	55.2	55.2	97.1	57.5	67.5
	2	40.4(−27%)	43.2(−22%)	36.6(−34%)	**108.7**	**60.6**	**70.0**
32	✗	110.5	110.5	110.5	49.6	60.5	69.4
	2	79.3(−28%)	89.5(−19%)	72.4(−34%)	**53.4**	**63.3**	**72.3**

schedule [33].

4.4 Results on Video Action Recognition

We first evaluate our method by applying it to R(2+1)D-18 [46] with different number of input frames and different size of search space. Here we use GFLOPs (floating point operations) to measure the computational cost of the model and report clip-1, video-1 and video-5 metrics to measure the accuracy of our models, where clip-1 is the top-1 accuracy of model evaluation with only one clip sampled from video, video-1 and video-5 are the top-1 and top-5 accuracy of model evaluated with *K-LeftCenterRight* strategy. Note that we report the FLOPs of a single video clips at the spatial resolution 256×256 (for I3D and X3D) or 128×128 (for R(2+1)D). In addition, we report the speed of each model with the metric of *clip/second*, which denotes the number of video clips that are processed in one second. We create the environment with PyTorch 1.6, CUDA 11.0, and a single NVIDIA TITAN RTX (24GB) GPU as our testbed to measure speed of different models. Table 4.1 shows the results (In all of the

Table 4.2: **Action recognition results on Mini-Kinetics-200.** We set the search space as 2 and train all the models with 16 frames. The metric speed uses *clip/second* as the unit.

Model	Dy.	GFLOPs	Speed	clip-1	video-1
R(2+1)D	✗	55.2	97.1	57.5	67.5
	✓	40.4	**108.7**	**60.6**	**70.0**
I3D	✗	56.0	116.4	59.7	68.3
	✓	26.5	**141.7**	**62.2**	**71.1**
X3D	✗	6.20	169.4	**66.5**	**72.2**
	✓	5.03	**178.2**	65.5	72.1

Table 4.3: **Action recognition results with Temporal Pyramid Network (TPN) on Mini-Kinetics-200.** TPN-8f and TPN-16f indicate that we use 8 frames and 16 frames as input to the model respectively.

Model	Dy.	GFLOPs	clip-1	video-1
TPN-8f	✗	28.5	58.9	67.2
	✓	21.5	**59.2**	**68.8**
TPN-16f	✗	56.8	59.8	68.5
	✓	41.5	**60.8**	**70.6**

tables, ✗ represents the original fixed model architecture while ✓ denote the dynamic model trained using our proposed approach). Our proposed approach VA-RED^2 significantly reduces the computational cost while improving the accuracy. We observe that dynamic model with the search space size of 2 has the best performance in terms of accuracy, GFLOPS and speed. We further test our VA-RED^2 with all of the three model architectures: R(2+1)D-18, I3D-InceptionV2, and X3D-M (Table 4.2) including the very recent temporal pyramid module [54] and correlation module [48] on Mini-Kinetics-200 dataset. We choose R(2+1)D-18 with TPN and CorrNet as the backbone architecture and test the performance of our method using a search space of 2 in Table 4.3 and Table 4.4 respectively. Table 4.2 shows that method boosts the speed of base I3D-InceptionV2 and R(2+1)D models by 21.7% and 10.6% respectively, showing its advantages not only in terms of GFLOPS but also in actual speed. Table 4.4 shows that our dynamic approach also outperforms the baseline CorrNet by 1.8% in top-1 video accuracy, while reducing the computational cost by

Table 4.4: **Comparison with CorrNet [48] and AR-Net [34] on Mini-Kinetics-200.** We set the search space as 2 and train all the models with 16 frames.

Model	Dy.	GFLOPS	clip-1	video-1	Method	Params	GFLOPs	clip-1
CorrNet	✗	60.8	59.9	68.2	AR-Net	63.0M	44.8	67.2
	✓	**45.5**	**60.4**	**70.0**	VA-RED^2	**23.9M**	**43.4**	**68.3**

Table 4.5: **Action recognition results on Kinetics-400.** We set the search space as 2, meaning models can choose to compute all feature maps or $\frac{1}{2}$ of them both on temporal and channel-wise convolutions. Here we set the number of frames as 16.

Model	Dynamic	16-frame				
		GFLOPs	speed	clip-1	video-1	video-5
R(2+1)D	✗	55.2	97.1	57.3	65.6	86.3
	✓	40.3	**105.9**	**58.4**	**67.6**	**87.6**
I3D	✗	56.0	116.4	55.1	66.5	86.7
	✓	32.1	**140.7**	**58.6**	**67.1**	**87.2**
X3D	✗	6.42	169.4	63.2	70.6	90.0
	✗	5.38	**177.6**	**65.3**	**72.4**	**90.7**

25.2% on Mini-Kinetics-200. Furthermore, we compare our method with AR-Net [34], which is a recent adaptive method that selects optimal input resolutions for video inference. We conduct our experiments on 16-frame TSN [49] with ResNet50 backbone and provide the comparison on FLOPs, parameter size, and accuracy (Table 4.4). To make a fair comparison, we train AR-Net using the official implementation on the same Mini-Kinetics-200 dataset with Kaiming initialization [19]. Table 4.4 shows that our method, VA-RED^2 outperforms AR-Net in both accuracy and GFLOPS, while using about 62% less parameters. Table 4.5, Table 4.6, and Table 4.7 show the results of different methods on Kinetics-400 and Moments-In-Time, respectively. To summarize, we observe that VA-RED^2 consistently improves the performance of all the base models including the recent architectures X3D, TPN, and CorrNet, while offering significant reduction in computation. Moreover, our approach is model-agnostic, which allows this to be served as a plugin operation for a wide range of action recognition architectures. From the comparison among different models, we find that our proposed VA-RED^2 achieves the most computation reduction on I3D-InceptionV2, between 40%

Table 4.6: **Action recognition results on Kinetics-400.** We set the search space as 2, meaning models can choose to compute all feature maps or $\frac{1}{2}$ of them both on temporal and channel-wise convolutions. Here we set the number of frames as 32.

Model	Dy.	32-frame				
		GFLOPs	speed	clip-1	video-1	video-5
R(2+1)D	✗	110.5	49.6	61.5	69.0	88.6
	✓	80.7	**53.0**	61.5	**70.0**	**88.9**
I3D	✗	112.0	57.6	57.2	64.9	86.5
	✓	64.3	**71.7**	**61.0**	**68.6**	**88.4**

Table 4.7: **Action recognition results on Moments-In-Time.** We set the search space as 2, i.e., models can choose to compute all feature maps or $\frac{1}{2}$ of them both on temporal and channel-wise convolutions. The speed uses *clip/second* as the unit.

Model	Dynamic	GFLOPs	speed	clip-1	video-1
R(2+1)D	✗	55.2	97.1	27.0	28.8
	✓	42.5	**105.5**	**27.3**	**30.1**
I3D	✗	56.0	116.4	25.7	26.8
	✓	32.1	**140.7**	**26.3**	**28.5**
X3D	✗	6.20	169.4	24.8	24.8
	✓	5.21	**177.4**	**26.7**	**27.7**

and 50%, while reducing less than 20% on X3D-M. This is because X3D-M is already very efficient both in terms of channel dimension and temporal dimension. Notice that the frames input to X3D-M are at the temporal stride of 5, which makes them share less similarity. Furthermore, we observe that dynamic I3D-InceptionV2 has very little variation of the computation for different input instances. This could be because of the topology configuration of the InceptionV2, which has lots of parallel structures inside the network architecture.

We also compare VA-RED^2 with a weight-level pruning method [17] and a automatic channel pruning method (CGNet) [23] on Mini-Kinetics-200. Table 4.8 shows that our approach significantly outperforms the weight-level pruning method by a margin of about 3%-4% in clip-1 accuracy with similar computation over the original fixed model and consistently outperforms CGNet while requiring less GFLOPs (maximum 2.8% in 16 frame). These results well demonstrate the effectiveness of our dynamic

Table 4.8: **Comparison with network pruning methods.** We choose R(2+1)D on Mini-Kinetics-200 dataset with different number of input frames. Numbers in green/blue quantitatively show how much our proposed method is better/worse than these pruning methods.

Method	Frames	GFLOPs	clip-1
Weight-level	8	19.9 **(-0.1)**	54.5 **(+3.2)**
	16	40.3 **(-0.1)**	57.7 **(+2.9)**
	32	79.6 **(-0.3)**	59.6 **(+3.7)**
CGNet	8	23.8 **(+3.8)**	56.2 **(+1.5)**
	16	47.6 **(+7.2)**	57.8 **(+2.8)**
	32	95.3 **(+16.0)**	61.8 **(+1.5)**

Table 4.9: **Action localization results on J-HMDB.** We set the search space as 2 for dynamic models. The speed uses *clip/second* as the unit.

Model	Dy.	GFLOPs	speed	mAP	Recall	Classif.
I3D	✗	43.9	141.1	44.8	**67.3**	87.2
	✓	21.3	**167.4**	**47.2**	65.6	**91.1**
X3D	✗	5.75	176.3	47.9	65.2	**93.2**
	✓	4.85	**184.6**	**50.0**	**65.8**	93.0

video redundancy framework over network pruning methods.

4.5 Results on Spatio-Temporal Action Localization

We further extend our method to the spatio-temporal action localization task to demonstrate the generalization ability to different task. We conduct our method on J-HMDB-21 with two different 3D backbone networks: I3D-InceptionV2 and X3D-M. We report frame-mAP at IOU threshold 0.5, recall value at IOU threshold 0.5, and classification accuracy of correctly localized detections to measure the performance of the detector. Table 4.9 shows that our dynamic approach outperforms the baselines on all three metrics while offering significant savings in FLOPs (e.g., more than 50% savings on I3D). In summary, VA-RED^2 is clearly better than the baseline architectures in terms of both accuracy and computation cost on both recognition and localization tasks, making it suitable for efficient video understanding.

Table 4.10: **Effect of efficiency loss on Kinetics-400.**

Model	*Efficency loss?*	GFLOPs	clip-1	video-1
R(2+1)D	No	49.8	57.9	66.7
	Yes	40.3	**58.4**	**67.6**
I3D	No	56.0	58.0	66.5
	Yes	32.1	**58.6**	**67.1**

Table 4.11: **Ablation experiments on dynamic modeling along temporal and channel dimensions.** We choose R(2+1)D-18 on Mini-Kinetics-200 and set the search space to 2 in all the dynamic models. Here we experiment with the 8-frame model.

Dynamic Temporal	Dynamic Channel	8-frame			
		GFLOPs	speed	clip-1	video-1
✗	✗	27.7	192.1	56.4	66.8
✓	✗	23.5	198.6	57.1	66.8
✗	✓	22.7	196.5	57.0	66.7
✓	✓	20.0	**205.5**	**57.7**	**68.0**

4.6 Effect of Efficiency Loss

We conduct an experiment by comparing the model performance before and after being finetuned with our proposed efficiency loss. Table 4.10 shows that finetuning our dynamic model with efficiency loss significantly reduces the computation without any accuracy loss.

4.7 Ablation Experiments on Dynamic Modeling

We test performance of or approach by turning of dynamic modeling along temporal and channel dimensions on Mini-Kinetics-200. Table 4.11 and Table 4.12 show that dynamic modeling along both dimensions obtains the best performance while requiring the least computation. This shows importance of input-dependent policy for deciding how many features need to be computed for both temporal and channel dimensions.

Table 4.12: **Ablation experiments on dynamic modeling along temporal and channel dimensions.** We choose R(2+1)D-18 on Mini-Kinetics-200 and set the search space to 2 in all the dynamic models. Here we experiment with the 16-frame model.

Dynamic Temporal	Dynamic Channel	16-frame			
		GFLOPs	speed	clip-1	video-1
✗	✗	55.2	97.1	57.5	67.5
✓	✗	46.1	105.0	58.6	67.6
✗	✓	46.3	102.0	59.2	68.3
✓	✓	40.4	**108.7**	**60.6**	**70.0**

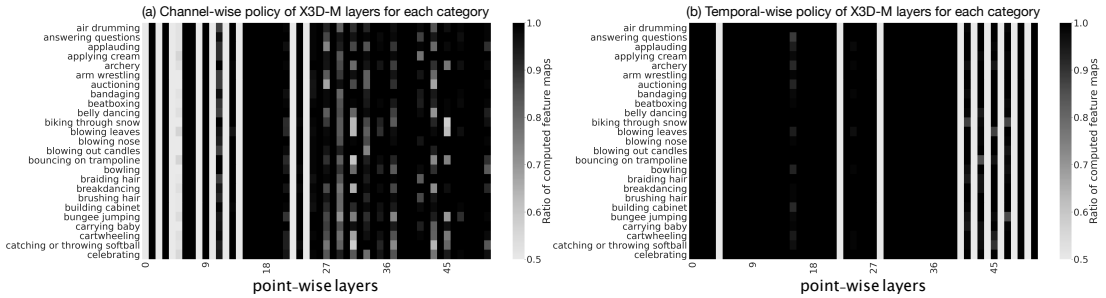


Figure 4-1: **Ratio of computed feature per layer and class on Mini-Kinetics-200 dataset.** We pick the first 25 classes of Mini-Kinetics-200 and visualize the per-block policy of X3D-M on each class. Lighter color means fewer feature maps are computed while darker color represents more feature maps are computed.

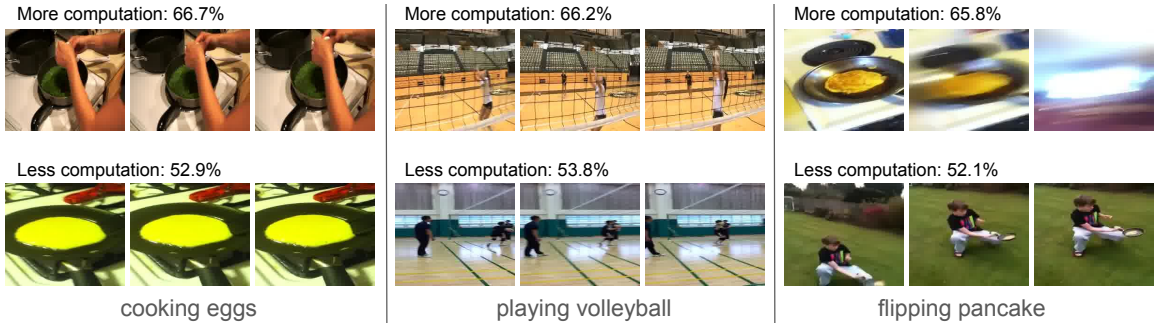


Figure 4-2: **Validation video clips from Mini-Kinetics-200.** For each category, we plot two input video clips which consume the most and the least computational cost respectively. We infer these video clips with 8-frame dynamic R(2+1)D-18 model trained on Mini-Kinetics-200 and the percentage indicates the ratio of actual computational cost of 2D convolution to that of the original fixed model. Best viewed in color.

4.8 Visualization and Analysis

To better understand the policy decision process, we dissect the network layers and count the ratio of feature maps that are being computed during each convolution layers for each category. From Figure 4-1, we observe that: In X3D, point-wise convolutions which right after the depth-wise convolutions have more variation among classes and network tends to consume more temporal-wise features at the early stage and compute more channel-wise features at the late stage of the architecture. The channel-wise policy has also more variation than the temporal-wise policy among different categories. Furthermore, we show few contrasting examples which are in the same category while requiring very different computation in Figure 4-2. Video clips which have more complicated scene configuration (e.g. cooking eggs and playing volleyball) and more violent camera motion (e.g. flipping pancake) tend to need more feature maps to do the correct predictions. More qualitative results can be found in Section B.3, Section B.4 and Section B.5 of the Appendix.

Chapter 5

Innovation and Intellectual Property

5.1 Context

The primary objective of this thesis is to explore the use of a dynamic network to dynamically allocate computational resources for processing different inputs, specifically focusing on video understanding tasks. In these tasks, computation can be highly resource-intensive. The proposed dynamic network aims to optimize computation allocation, resulting in improved performance without compromising the quality of the results.

The core idea of this thesis has led to the development of a patented method [38] (ADAPTIVE REDUNDANCY REDUCTION FOR EFFICIENT VIDEO UNDERSTANDING, United States Patent Application 20230082448, 09/15/2021, Pan, Bowen, Panda, Rameswar, Fosco, Camilo Luciano, Feris, Rogerio Schmidt, Oliva, Aude Jeanne) for improving the performance of a computer that uses a convolutional neural network (CNN) to carry out video processing tasks. The method revolves around applying an input-dependent policy network for each convolution layer in the CNN to determine the fraction of input feature maps to be fully computed and the fraction to be reconstructed. This dynamic approach to computation allocation allows for efficient processing while maintaining the quality of the output.

The patented method consists of several claims that detail the process of applying the input-dependent policy network, the determination of fractions based on redundant

and non-redundant feature maps, and the simultaneous joint training of the CNN and the policy network. Furthermore, the method also considers the dynamic scaling of temporal strides and the number of output channels based on the output of the policy network.

This thesis investigates the implications of incorporating the patented method in video understanding tasks, such as video recognition, spatio-temporal action localization, and video segmentation. The goal is to assess the effectiveness of the dynamic network in optimizing computational resources for different input types, ultimately enhancing the performance and efficiency of video processing tasks. The successful application of the patented method based on this research will serve as a foundation for exploring the potential of dynamic computation allocation in various video understanding tasks and applications.

5.2 Introduction to the Patent

The present patent [38], titled "Adaptive Redundancy Reduction for Efficient Video Understanding" (United States Patent Application 20230082448), focuses on a novel method that improves the performance of a computer utilizing a convolutional neural network (CNN) to carry out video processing tasks. This innovative method, developed by inventors Pan, Panda, Fosco, Feris, and Oliva, is designed to optimize the efficiency of CNNs by applying an input-dependent policy network to each convolution layer.

The input-dependent policy network helps to determine two fractions of input feature maps for each convolution layer: the first fraction, whose corresponding output feature maps are to be fully computed, and the second fraction, which will be reconstructed from the first corresponding output feature maps instead of being fully computed. This approach enables the CNN to prioritize non-redundant feature maps, reducing computational overhead without sacrificing the quality of the output.

The patented method also encompasses the simultaneous joint training of the CNN and the policy network, allowing for dynamic scaling of temporal strides and the number of output channels. This results in a more efficient and optimized network for video

understanding tasks such as video recognition, spatio-temporal action localization, and video segmentation.

5.3 Explanation of the Technology

The technology described in the patent involves adaptive redundancy reduction for efficient video understanding using convolutional neural networks (CNNs). It addresses the challenge of performing inference on deep learning models for videos, which requires a large amount of computational resources for robust recognition. Real-world videos have a high correlation of information across frames, leading to redundancy in temporal or spatial feature maps, or both. Static videos tend to have more temporal redundancy, while videos focusing on objects often have more channel redundancy.

The invention offers techniques to improve the performance of a computer using a CNN for video processing tasks. It does so by applying an input-dependent policy network for each convolution layer in the CNN. The policy network determines the following:

- A first fraction of input feature maps for which the corresponding output feature maps are fully computed by the given convolution layer.
- A second fraction of input feature maps for which the corresponding output feature maps are not fully computed by the given convolution layer but are reconstructed from the first corresponding output feature maps.

For each convolution layer in the CNN, the method involves fully computing the first corresponding output feature maps from the first fraction of input feature maps and reconstructing the second corresponding output feature maps from the first corresponding output feature maps. Finally, for the last convolution layer, the first and second corresponding output feature maps are input to an output layer to obtain an inference result.

The apparatus implementing this method includes a memory embodying computer-executable instructions and at least one processor coupled to the memory. The

processor operates by executing the instructions to perform the method. The method involves instantiating a CNN and an input-dependent policy network, applying the policy network for each convolution layer, fully computing the first corresponding output feature maps, reconstructing the second corresponding output feature maps, and obtaining an inference result.

The technology provides substantial beneficial technical effects, such as reducing central processing unit (CPU) and memory requirements and reducing runtime for video processing tasks using neural networks on a computer. It enhances efficiency by exploiting temporal and channel redundancy in videos. The accompanying drawings illustrate a framework and a system for adaptive temporal and channel redundancy reduction, as well as dynamic convolution along temporal and channel dimensions, action recognition results, visualizations of temporal-wise and channel-wise feature maps, and policy visualizations for learning different network layers.

More details can be referred to the patent [38].

5.4 Potential Impact

The technology of adaptive redundancy reduction for efficient video understanding using convolutional neural networks has both positive and negative potential impacts:

Positive Impacts:

- Improved efficiency: By exploiting temporal and channel redundancy in videos, this technology can greatly improve the efficiency of video processing tasks using neural networks, resulting in faster processing times and reduced computational costs.

- Reduced resource requirements: The technology can significantly reduce the CPU and memory requirements for video processing tasks, making it more accessible to users with limited resources, such as those using consumer-grade hardware.

- Enhanced video understanding: With more efficient video processing, this

technology can lead to more accurate and robust video recognition, benefiting various applications like video surveillance, autonomous vehicles, robotics, and content analysis.

- Energy savings: Reducing computational complexity and resource requirements can lead to energy savings, which is essential for reducing the environmental impact of computing technologies and enabling their use in battery-powered devices.

- Scalability: The technology enables the processing of large-scale video datasets or real-time video streams with minimal resource requirements, facilitating the development and deployment of scalable video analysis solutions.

Negative Impacts:

- Over-reliance on automation: As video understanding technology becomes more efficient, there could be an over-reliance on automation, leading to reduced human involvement in video analysis tasks. This may result in a lack of critical thinking or oversight in certain situations where human intervention is necessary.

- Privacy concerns: Enhanced video understanding technology could potentially be misused for mass surveillance or invading people's privacy. With more efficient video processing, it could become easier for governments or organizations to analyze and monitor video feeds without consent, leading to potential privacy violations.

- Unemployment: The increased efficiency and automation in video processing tasks may lead to job displacement for people working in fields related to video analysis, as machines become capable of performing these tasks more effectively and efficiently.

- Bias and discrimination: If not carefully designed, the technology could perpetuate or exacerbate biases present in the training data, leading to unfair or discriminatory outcomes in video analysis tasks.

- Misuse: The technology can be used for malicious purposes, such as creating deepfakes or manipulating video content to spread misinformation or cause harm.

The overall impact of this technology depends on its implementation, regulation, and responsible use. Ensuring transparency, addressing ethical concerns, and developing guidelines for the technology's application can help mitigate the negative impacts and maximize its positive contributions to society.

Appendix A

Technical Details

A.1 Dataset Details

We evaluate the performance of our approach using three video action recognition datasets, namely Mini-Kinetics-200 [34], Kinetics-400 [4], and Moments-In-Time [36] and one spatio-temporal action localization task namely J-HMDB-21 [25]. Kinetics-400 is a large dataset containing 400 action classes and 240K training videos that are collected from YouTube. The Mini-Kinetics dataset contains 121K videos for training and 10K videos for testing, with each video lasting 6-10 seconds. The original Kinetics dataset is publicly available to download at `https://deepmind.com/research/open-source/kinetics`. We use the official training/validation/testing splits of Kinetics-400 and the splits released by authors in [34] for Mini-Kinetics-200 in our experiments.

Moments-in-time [36] is a recent collection of one million labeled videos, involving actions from people, animals, objects or natural phenomena. It has 339 classes and each video clip is trimmed to 3 seconds long. This dataset is designed to have a very large set of both inter-class and intra-class variation that captures a dynamic event at different levels of abstraction (i.e. "opening" doors, curtains, mouths, even a flower opening its petals). We use the official splits in our experiments. The dataset is publicly available to download at `http://moments.csail.mit.edu/`.

Joints for the HMDB dataset (J-HMDB-21 [25]) is based on 928 clips from HMDB51

comprising 21 action categories. Each frame has a 2D pose annotation based on a 2D articulated human puppet model that provides scale, pose, segmentation, coarse viewpoint, and dense optical flow for the humans in action. The 21 categories are brush hair, catch, clap, climb stairs, golf, jump, kick ball, pick, pour, pull-up, push, run, shoot ball, shoot bow, shoot gun, sit, stand, swing baseball, throw, walk, wave. The dataset is available to download at http://jhmdb.is.tue.mpg.de/.

A.2 Implementation Details

Details of Shared-weight Training and Inference. In this section, we provide more details of the shared-weight mechanism presented in Section 3 of the main paper. We first compute all the possible necessary features using a big kernel and then for each dynamic convolution with different scaling factor, we sample its corresponding ratio of necessary features and reconstruct the rest features by cheap operations to get the final output. For example, the original channel-wise dynamic convolution at ratio $r = (\frac{1}{2})^{(i-1)}$ can be analogized to

$$\big[(f_l^c(X_l, r = (\frac{1}{2})^{i_s^c-1})[0 : (\frac{1}{2})^{(i-1)}C_{out}]), (\Phi^c(f_l^c(X_l, r = (\frac{1}{2})^{i_s^c-1})[0 : (\frac{1}{2})^{(i-1)} \cdot C_{out}]))\big], \tag{A.1}$$

where $[\cdot : \cdot]$ is the index operation along the channel dimension, and i_s^c is the index of the largest channel-wise filter, during training phase, we have $i_s^c = 1$, while during inference phase, i_s^c is the smallest index for V_c^l, $s.t. V_c^l[i_s^c] = 0$. By utilizing such a share-weight mechanism, the computation of the total channel-wise dynamic convolution is reduce to $(\frac{1}{2})^{i_s^c-1} \cdot \mathcal{C}(f_l)$. Further, we have the total computational cost of the adjunct process as

$$\mathcal{C}(f_l^{t,c}) = (\frac{1}{2})^{i_s^c+i_s^t-2} \cdot \mathcal{C}(f_l), \tag{A.2}$$

where i_s^t is the index of largest temporal-wise filter.

Appendix B

Analysis

Training and Inference. We apply our method mainly to 2D convolutions in R(2+1)D since 2D convolution takes the most computational cost compared with 1D convolution. We train most of our models on 96 NVIDIA Tesla V100-32GB GPUs and perform synchronized BN [24] across all the GPUs. For R(2+1)D [46], the learning rate is initialized as 0.18 and the weight decay is set to be 5×10^{-4}. For I3D [4, 53] and X3D [9], the learning rates both start from 1.8 and weight decay factors are 1×10^{-4} and 5×10^{-5} respectively. Cosine learning rate decaying strategy is applied to decrease the total learning rate. All of the models are trained from scratch and warmed up for 15 epochs on mini-Kinetics/Kinetics, 8 epochs on Moments-In-Time dataset. We adopt the Nesterov momentum optimizer with an initial weight of 0.01 and a momentum of 0.9. During training, we follow the data augmentation (location jittering, horizontal flipping, corner cropping, and scale jittering) used in TSN [49] to augment the video with different sizes spatially and flip the video horizontally with 50% probability. We use single-clip, center-crop FLOPs as a basic unit of computational cost. Inference-time computational cost is roughly proportional to this, if a fixed number of clips and crops is used, as is for our all models. Note that Kinetics-400 dataset is shrinking in size (∼15% videos removed from original Kinetics) and the original version used in [4] are no longer available from official site, resulting in some difference of results.

Table B.1: **Quantitative results of redundancy experiments.** We compute the correlation coefficient, RMSE and redundancy proportions (RP) for feature maps in well-known pretrained video models on Moments-in-Time and Kinetics-400 datasets. RP is calculated as the number of tensors with both CC and RMSE above redundancy thresholds of 0.85 and 0.001, respectively. We show results corresponding to averaging the per layer values for all videos in the validation sets. We observe that networks trained on Moments-In-Time (and evaluated on the Moments in Time validation set) tend to present slightly less redundancy than their Kinetics counterparts, and the time dimension tends to be more redundant than the channel dimension in all cases. We observe severe redundancy across the board (with some dataset-model pairs achieving upwards of 0.8 correlation coefficient between their feature maps), which further motivates our redundancy reduction approach.

Dataset	Model	Dimension	CC	RMSE	RP
Moments-In-Time	I3D	Temporal	0.77	0.083	0.62
	I3D	Channel	0.71	0.112	0.48
	R(2+1)D	Temporal	0.73	0.108	0.49
	R(2+1)D	Channel	0.68	0.122	0.43
Kinetics-400	I3D	Temporal	0.81	0.074	0.68
	I3D	Channel	0.76	0.091	0.61
	R(2+1)D	Temporal	0.78	0.081	0.64
	R(2+1)D	Channel	0.73	0.088	0.58

B.1 Redundancy Analysis

To motivate our redundancy reduction approach, we measure and visualize the internal redundancy of well known pretrained networks. We analyze the internal feature maps of existing pre-trained I3D-InceptionV2 and R(2+1)D networks on Moments in Time and Kinetics. For each model-dataset pair, we extract feature maps for all examples in the validation sets, in both time and channel dimensions, and measure their similarity. In detail, our method consists of the following steps: (1) For a given input, we first extract the output feature maps from all convolutional layers in the network at hand. (2) In each layer, we measure the similarity of each feature map to each other with Person's correlation coefficient (CC) and root mean squared error (RMSE). We additionally flag feature maps that exhibit high similarity as redundant. (3) After computing this for the validation sets, we average the values over all examples to obtain mean metrics of redundancy per model and per dataset. We additionally

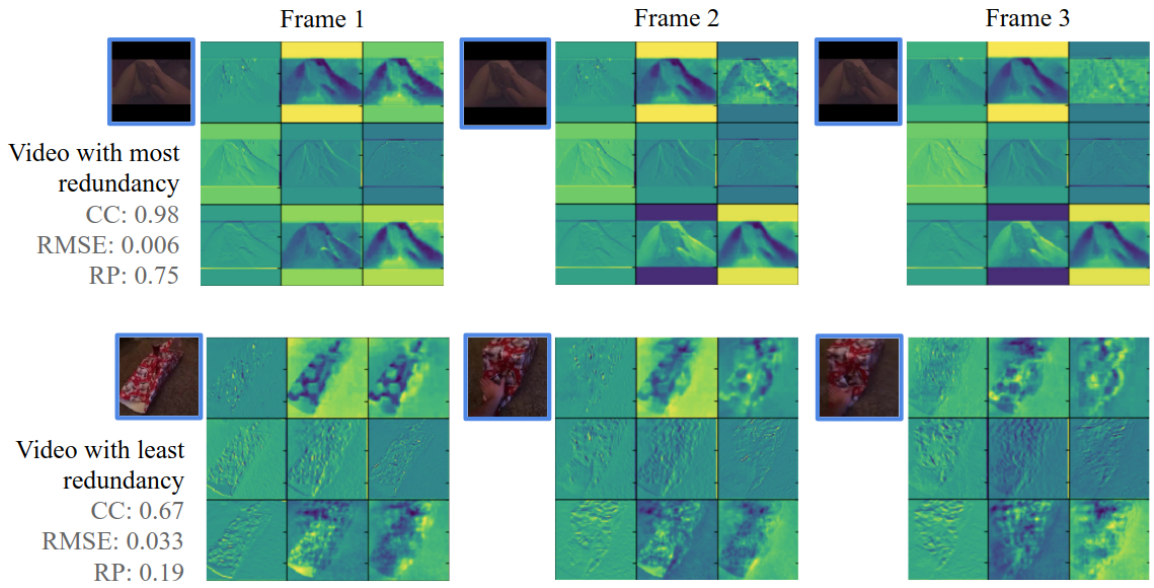


Figure B-1: Visualization of the first 9 filters of the first layer of I3D, on examples with most (top) and least (bottom) redundancy in the temporal dimension. We exemplify the results on frames 1, 2 and 3. As can be seen, the video with most redundancy consists of a relatively static video with little movement, and the sets of feature maps from frame to frame harbor heavy similarity. The video with least redundancy consists of a gift unwrapping with rapid movement (even in the first few frames) and the corresponding feature maps present visible structural differences from frame to frame. Although in both cases, redundancy is present, it is clear that some examples present much more redundancy than others, thus motivating our input-dependent redundancy reduction approach.

compute the ranges of these values to visualize how much redundancy can vary in a model-dataset pair. We present quantitative results in Table B.1 and show examples of our findings in Figure B-1.

B.2 VA-RED^2 on Longer-training Model

In our experiments, all of our models are trained under a common evaluation protocol for a fair comparison. To balance the training cost and model performance, we use a smaller epoch size than the original paper to train our models. For example, authors in [46] and [9], train the R(2+1)D models and X3D models for 188 epochs and 256 epochs respectively to pursue the state-of-the art. However, we only train the models for 120 epochs to largely save the computation resources and training time. However, to rule

out the possibility that our base models (i.e., without using Dynamic Convolution) benefit from longer training epochs while our VA-RED^2 may not, we conduct an ablation study on the epoch size in Table B.2. We can see that our method still shows superiority over the base model in terms of the computational cost and accuracy on the 256-epoch model. Thus we conclude that the effectiveness of our method in achieving higher performance with low computation also holds on the longer-training models.

Table B.2: **Comparison between the performance of VA-RED^2 on 120-epoch X3D model and 256-epoch X3D model.** We choose X3D-M as our backbone architecture and set the search space as 2. We train one group of models for 120 epochs and the other for 256 epochs.

Model	Dynamic	120 epochs			256 epochs		
		GFLOPs	clip-1	video-1	GFLOPs	clip-1	video-1
X3D-M	✗	6.42	63.2	70.6	6.42	64.4	72.3
	✓	5.38	**65.3**	**72.4**	5.87	**66.4**	**73.6**

B.3 Feature Map Visualizations

To further validate our initial motivation, we visualize the feature maps which are fully computed by the original convlution operation and those which are generated by the cheap operations. We demonstrate those in both temporal dimension (c.f. Figure B-2) and channel dimension (c.f. Figure B-3). In both cases we can see that the proposed cheap operation generates meaningful feature maps and some of them looks even no difference from the original feature maps.

B.4 Policy Visualizations

To compare with the policy on Mini-Kinetics-200 (Figure 3 of the main paper), we also visualize the ratio of features which are consumed in each layer on Kinetics-400 (c.f. Figure B-4) and Moments-In-Time (c.f. Figure B-5). We can see from these two figures that the conclusions we draw from Mini-Kientics-200 still hold. Specifically, In

X3D, point-wise convolutions which right after the depth-wise convolutions have more variation among classes and network tends to consume more temporal-wise features at the early stage and compute more channel-wise features at the late stage of the architecture. However, R(2+1)D choose to select fewer features at early stage by both temporal-wise and channel-wise policy. Furthermore, we count the FLOPs of each instance on Mini-Kinetics-200, Kinetics-400, and Moments-In-Time and plot pie charts to visualize the the distribution of this instance-level computational cost. We analyze such distribution with two models: R(2+1)D-18 and X3D-M. All of the results are demonstrated in Figure B-6.

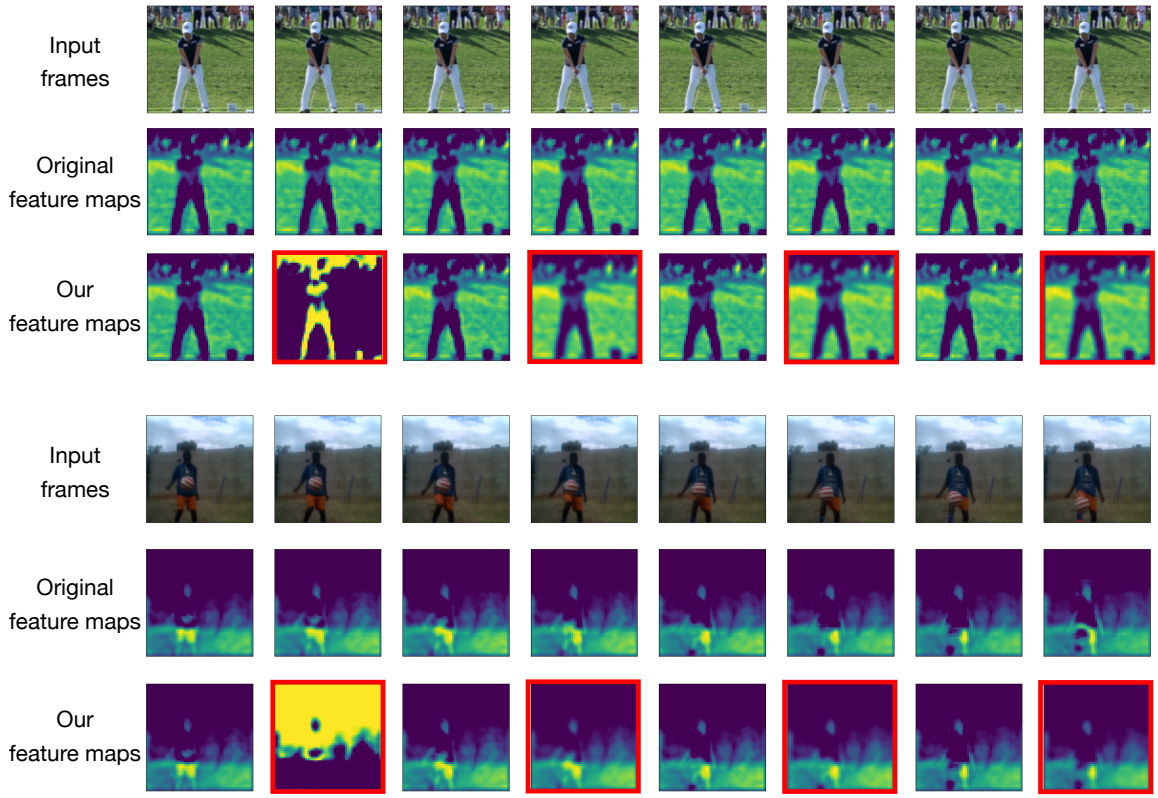


Figure B-2: **Visualization of temporal-wise feature maps.** We plot the temporal feature maps which are fully computed by the original convolution and those mixed with cheaply generated feature maps. The feature maps marked with red bounding boxes are cheaply generated. We do this analysis on 8-frame dynamic R(2+1)D-18 pretrained on Mini-Kinetics-200. These feature maps are the output of the first spatial convolution combined with ReLU non-linearity inside the `ResBlock_1`. We can see that most of the cheaply generated feature maps looks no difference from the original feature maps, which further support our approach. Best viewed in color.

B.5 Qualitative Results

We show additional input examples which consume different levels of computational cost on Kinetics-400 dataset (c.f. Figure B-7) and Moments-In-Time dataset (c.f. Figure B-8). To be consistent, we use the 16-frame dynamic R(2+1)D-18 as our pre-trained model. We can see that the examples consuming less computation tend to have less temporal motion, like the second example in Figure B-7, or have a relatively simple scene configuration, like the first and second examples in Figure B-8.

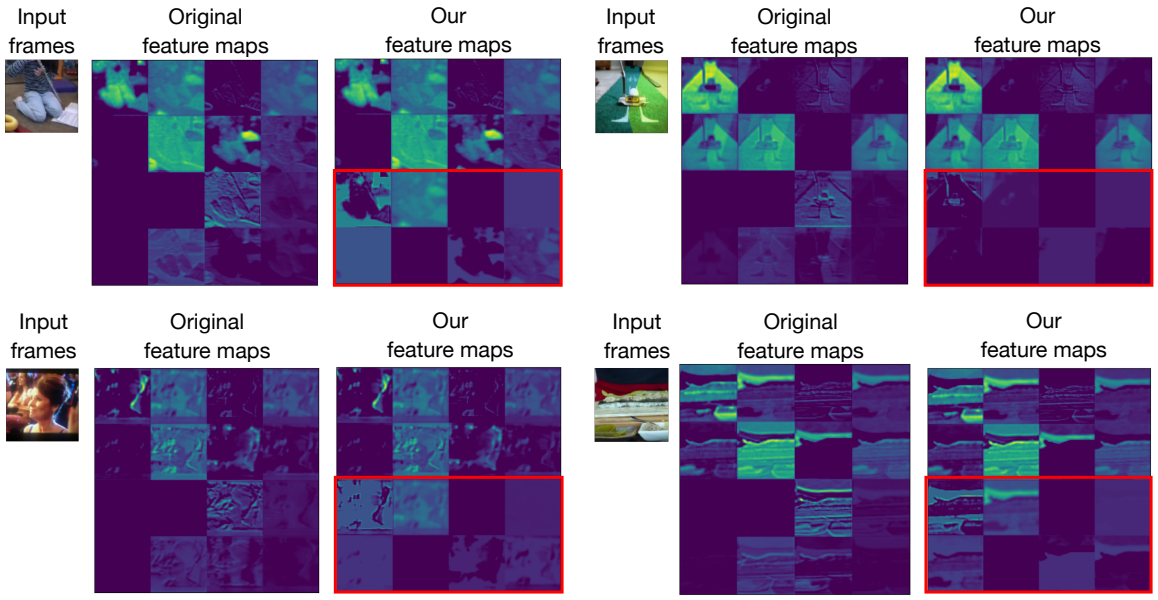


Figure B-3: **Visualization of channel-wise feature maps.** We plot the feature maps across the channel dimension. We contrast two kinds of feature maps: fully computed by the original convolution and those mixed with cheaply generated feature maps. The feature maps inside the red bounding boxes are cheaply generated. The analysis is performed on 8-frame dynamic R(2+1)D-18 model which is pretrained on Mini-Kinetics-200 dataset and we extract these feature maps which are output by the first spatial convolution layer inside the `ResBlock_1`. Best viewed in color.

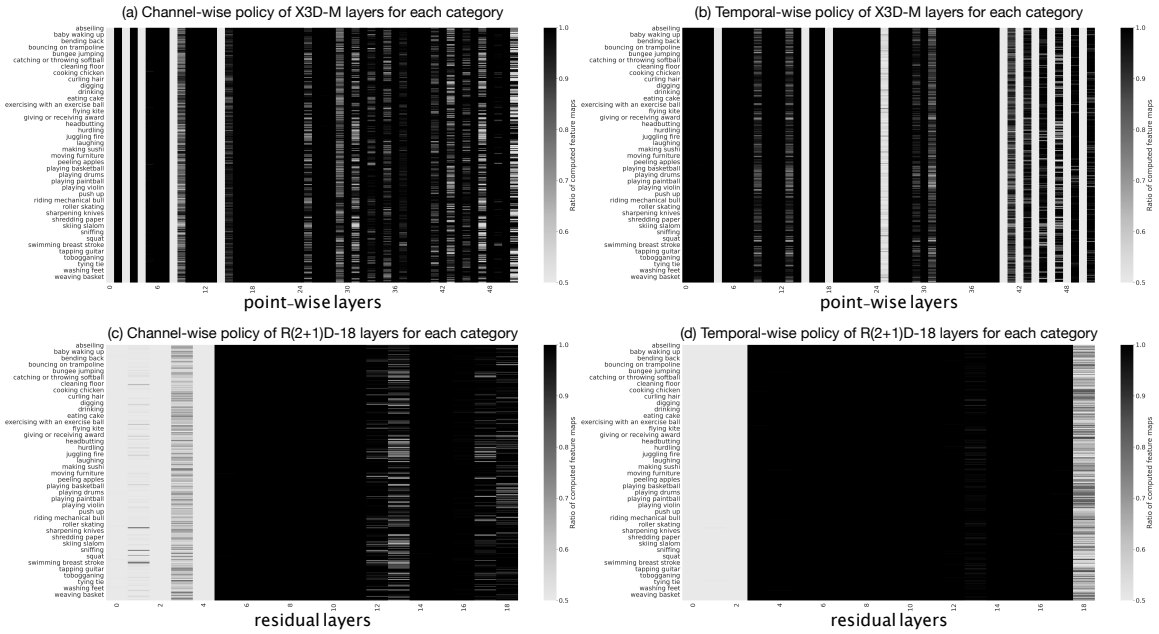


Figure B-4: **Ratio of computed feature per layer and class on Kinetics-400 dataset.** We visualize the per-block policy of X3D-M and R(2+1)D-18 on all 400 classes. Lighter color means fewer feature maps are computed while darker color represents more feature maps are computed. While X3D-M tends to consume more temporal-wise features at the early stage and compute more channel-wise features at the late stage, R(2+1)D choose to select fewer features at early stage by both temporal-wise and channel-wise policy. For both architectures, the channel-wise policy has more variation than the temporal-wise policy among different categories.

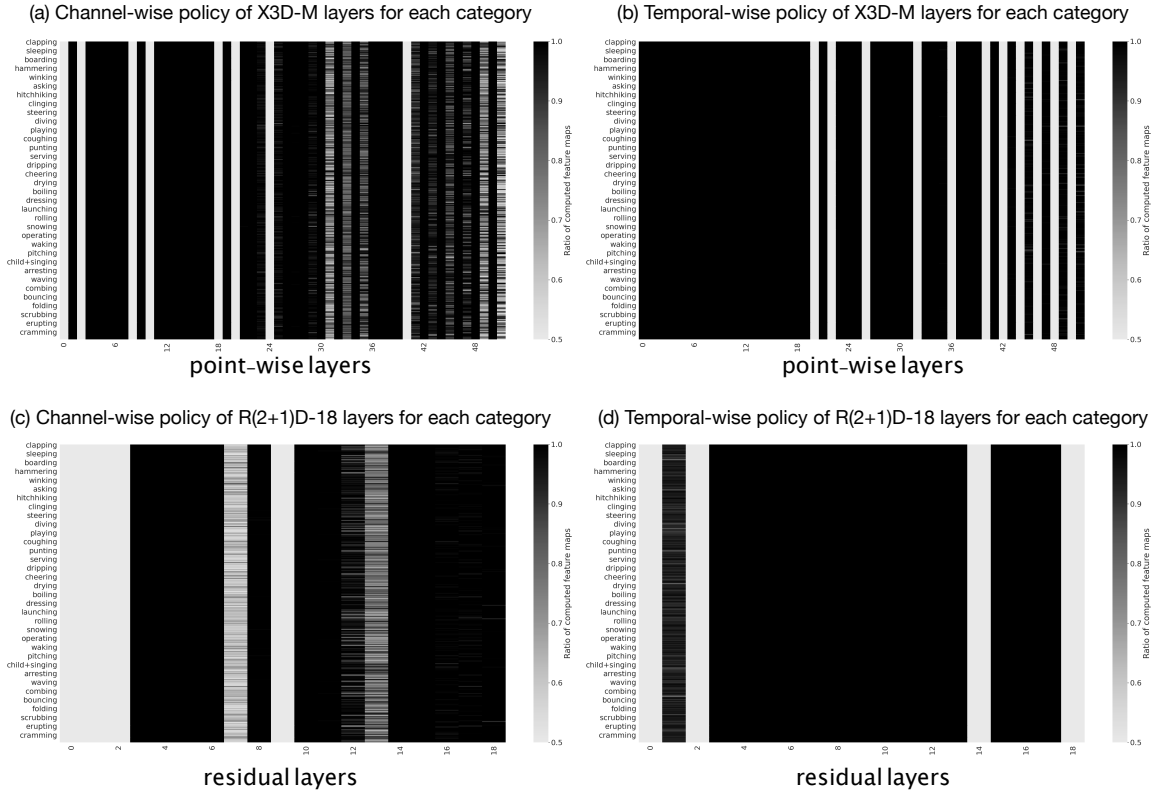


Figure B-5: **Ratio of computed feature per layer and class on Moments-In-Time dataset.** We visualize the per-block policy of X3D-M and R(2+1)D-18 on all 339 classes. Lighter color means fewer feature maps are computed while darker color represents more feature maps are computed.

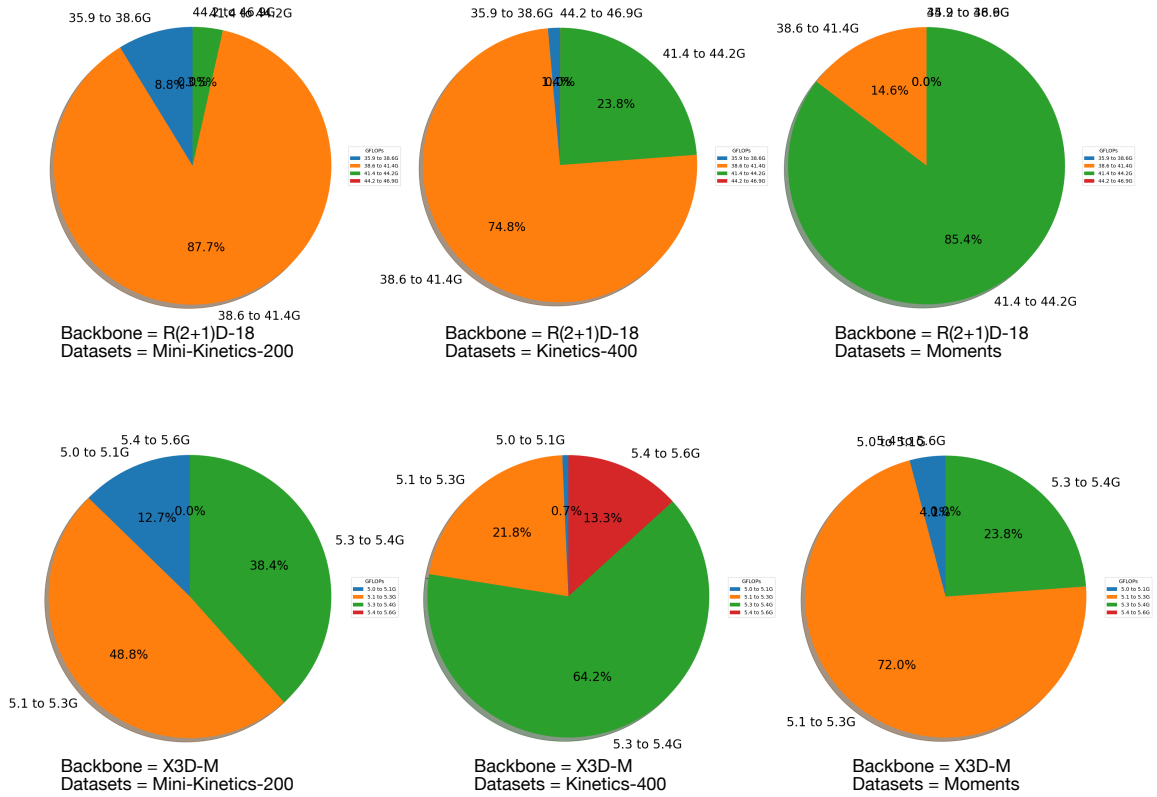


Figure B-6: **Computational cost distribution across different models on different datasets.** We count the computation of each instance cost by different models on different datasets. For instance, for the upper-left one, we use the model backbone of R(2+1)D-18 on Mini-Kinetics-200. This sub-figure indicates that there are 87.7% of videos in Mini-Kinetics-200 (Dataset) consuming $38.6 - 41.4$ GFLOPs by using R(2+1)D-18 (Backbone), 8.8% of videos consuming $35.9 - 38.6$ GFLOPs, and 3.5% of videos consuming $41.4 - 44.2$ GFLOPs.

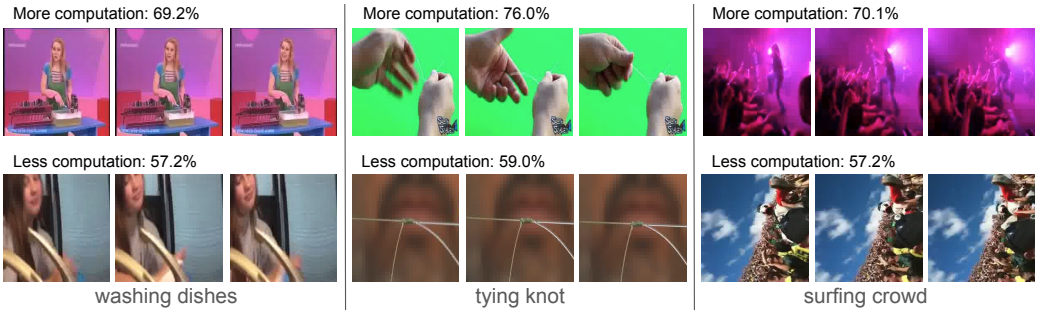


Figure B-7: **Validation video clips from Kinetics-400.** For each category, we plot two input video clips which consume the most and the least computational cost respectively. We infer these video clips with 16-frame dynamic R(2+1)D-18 which is pre-trained on Kinetics-400. The percentage in the figure indicates the ratio of the actual computational cost of 2D convolution to that of the original fixed model. Best viewed in color.

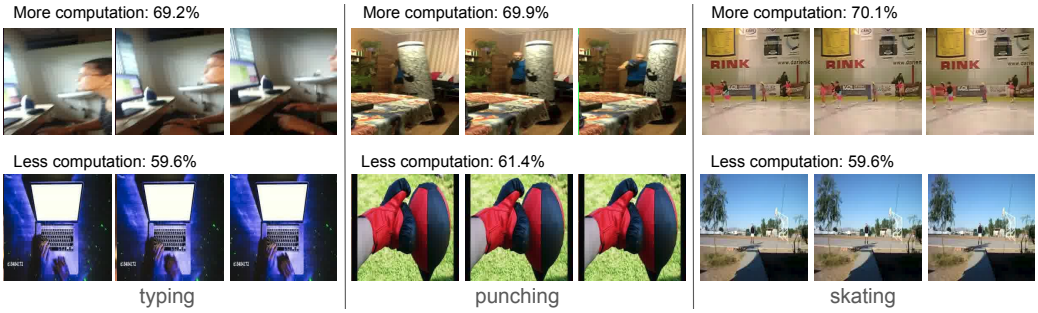


Figure B-8: **Validation video clips from Moments-In-Time.** For each category, we plot two input video clips which consume the most and the least computational cost respectively. We infer these video clips with 16-frame dynamic R(2+1)D-18 which is pre-trained on Moments-In-Time. The percentage in the figure indicates the ratio of the actual computational cost of 2D convolution to that of the original fixed model. Best viewed in color.

Bibliography

[1] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[4] 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.

[5] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015.

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

[7] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *European Conference on Computer Vision*, pages 505–521. Springer, 2020.

[8] Quanfu Fan, Chun-Fu Richard Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *Advances in Neural Information Processing Systems*, pages 2261–2270, 2019.

[9] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. *arXiv preprint arXiv:2004.04730*, 2020.

[10] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2018.

[11] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777, 2017.

[12] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. *arXiv preprint arXiv:1912.04487*, 2019.

[13] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768, 2015.

[14] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[15] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4805–4814, 2019.

[16] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[17] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[18] 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.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[20] 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.

[21] 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.

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[23] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. In *Advances in Neural Information Processing Systems*, pages 1886–1896, 2019.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3192–3199, 2013.

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

[27] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. 2019.

[28] 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.

[29] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[30] Xinyu Li, Bing Shuai, and Joseph Tighe. Directional temporal modeling for action recognition. In *European Conference on Computer Vision*, pages 275–291. Springer, 2020.

[31] 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.

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[33] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[34] 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. 2020.

[35] Yue Meng, Rameswar Panda, Chung-Ching Lin, Prasanna Sattigeri, Leonid Karlinsky, Kate Saenko, Aude Oliva, and Rogerio Feris. Adafuse: Adaptive temporal fusion network for efficient action recognition. In *International Conference on Learning Representations*, 2021.

[36] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.

[37] Bowen Pan, Wuwei Lin, Xiaolin Fang, Chaoqin Huang, Bolei Zhou, and Cewu Lu. Recurrent residual module for fast inference in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Bowen Pan, Rameswar Panda, Camilo Luciano Fosco, Rogerio Schmidt Feris, and Aude Jeanne Oliva. Adaptive redundancy reduction for efficient video understanding, March 16 2023. US Patent App. 17/476,437.

[39] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[41] 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.

[42] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[43] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[44] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[45] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

[46] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In

Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 6450–6459, 2018.

[47] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[48] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 352–361, 2020.

[49] 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. Springer, 2016.

[50] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[51] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[52] 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 Conference on Computer Vision and Pattern Recognition*, pages 1278–1287, 2019.

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

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

[55] 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.

[56] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.

[57] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[59] 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.

[60] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[61] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.

[62] Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. Mict: Mixed 3d/2d convolutional tube for human action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 449–458, 2018.

[63] 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.