Appendix

A. Datasets and Baselines

A.1. Datasets

Our experiments are based on six widely-used large-scale video recognition datasets. For all of them, we use the official training-validation split.

- ActivityNet [2] contains 10,024 training videos and 4,926 validation videos sorted into 200 human action categories. The average duration is 117 seconds.
- FCVID [33] contains 45,611 videos for training and 45,612 videos for validation, which are annotated into 239 classes. The average duration is 167 seconds.
- Mini-Kinetics is a subset of the Kinetics [34] dataset. We establish it following [21, 35, 47, 48, 57, 69]. The dataset include 200 classes of videos, 121k for training and 10k for validation. The average duration is around 10 seconds [34].
- Something-Something (Sth-Sth) V1&V2 [23] datasets include 98k and 194k videos respectively. Both of them are labeled with 174 human action classes. The average duration is 4.03 seconds.
- Jester [45] dataset consists of 148,092 videos in 27 action categories. The average duration is 3 seconds.

Data pre-processing. Following [40,47,64], the training data is augmented via random scaling followed by 224x224 random cropping, after which random flipping is performed on all datasets except for Sth-Sth V1&V2 and Jester. At test time, since we consider improving the inference efficiency of video recognition, we resize the short side of video frames to 256 and perform 224x224 centre-crop, obtaining a single clip per video for evaluation.

A.2. Baselines

In addition to AdaFocusV1, our proposed AdaFocusV2 approach is compared with several state-of-the-art frameworks designed for efficient video recognition, including MultiAgent [67], LiteEval [69], SCSampler [36], ListenToLook [19], AR-Net [47], AdaFrame [68], AdaFuse [48], VideoIQ [57], Dynamic-STE [35] and FrameExit [21]. Here we briefly introduce them.

- MultiAgent [67] learns to attend to important frames using multi-agent reinforcement learning. The implementation in [47] is adopted.
- LiteEval [69] allocates computation dynamically according to the importance of frames by switching between coarse and fine LSTM networks.
- SCSampler [36] is an efficient framework to select salient temporal clips from a long video. The implementation in [47] is adopted.
- ListenToLook [19] selects the key clips of a video by leveraging audio information. We adopt the image-based variant introduced in their paper for fair comparisons, since we do not use the audio of videos.
- AR-Net [47] processes video frames with different resolutions based on their relative importance.
- AdaFrame [68] learns to adaptively identify informative frames on a per-video basis with reinforcement learning. Each video is processed using different numbers of frames, facilitating dynamic inference.
- AdaFuse [48] proposes to dynamically fuse channels along the temporal dimension for modeling temporal relationships effectively.
- VideoIQ [57] learns to select optimal precision for each frame conditioned on their importance in terms of video recognition.
- Dynamic-STE [35] adopts a lighter student network and a heavier teacher network to process more and less frames, respectively. The two networks dynamically interact with each other during inference.
- FrameExit [21] learns to process relatively fewer frames for simpler videos and more frames for difficult ones.

B. Implementation Details

B.1. Architecture of the Policy Network π

We follow the design of π adopted by AdaFocusV1 [64]. The global feature maps $\boldsymbol{e}_t^{\mathrm{G}}$ of each frame is compressed to 64 channels by a 1x1 convolutional layer, vectorized, and fed into a one-layer gated recurrent unit (GRU) [6] with a hidden size of 2048. The outputs are projected to 2 dimensions (*i.e.*, $(\tilde{x}_{\mathrm{c}}^t, \tilde{y}_{\mathrm{c}}^t)$) and processed by the sigmoid activation function. On top of TSM, since a single patch location is generated for each video, we concatenate $\boldsymbol{e}_t^{\mathrm{G}}$ of all frames as the input of π, and replace the GRU by an MLP.

B.2. Training Hyper-parameters

In general, we find that the performance of AdaFocusV2 does not rely on the extensive hyper-parameter searching on a dataset or patch size basis. The training hyper-parameters only need to be tuned when the backbone networks (*i.e.*, f_{G} and f_{L}) change, and it may largely follow the training protocol for solely training the backbones (*e.g.*, typically, this can be easily obtained from the official implementation in the literature).

Table 7. **Effectiveness of the learned patch selection policy.**

Policy (128^2 patches)	ActivityNet mAP after processing t frames (*i.e.*, corresponding to $\boldsymbol{p}_t$)				
	t=1	t=2	t=4	t=8	t=16
Random Policy	37.7%	46.0%	56.8%	67.6%	73.9%
Central Policy	39.8%	47.7%	57.3%	66.6%	72.6%
Gaussian Policy	35.6%	44.5%	55.5%	67.1%	73.4%
AdaFocusV2-MN2/RN	**44.8%**	**52.5%**	**61.7%**	**71.0%**	**76.4%**

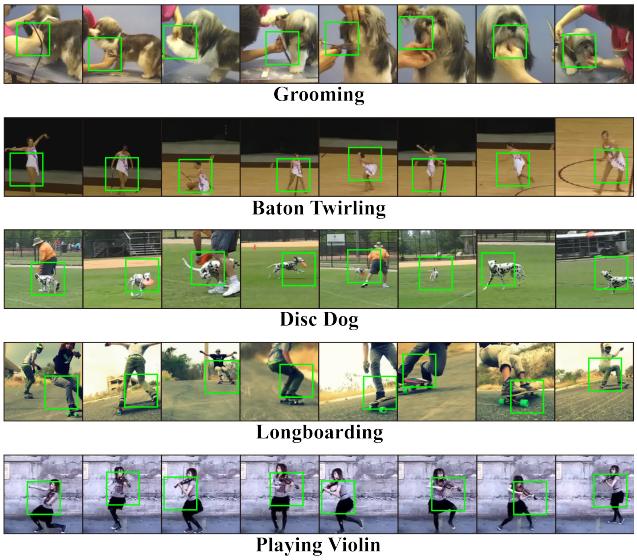

Figure 10. **Visualization results (zoom in for details).**

ActivityNet, FCVID and Mini-Kinetics (Section 4.1). As stated in the paper, all the components (*i.e.*, f_{G}, f_{L}, f_{C} and π) of AdaFocusV2 are trained simultaneously in a standard end-to-end fashion. An SGD optimizer with cosine learning rate annealing and a momentum of 0.9 is adopted. The L2 regularization co-efficient is set to 1e-4. The two encoders f_{G} and f_{L} are initialized using the ImageNet pre-trained models[3] , while f_{C} and π are trained from random initialization. On ActivityNet and FCVID, the size of the mini-batch is set to 32. The initial learning rates of f_{G}, f_{L}, f_{C} and π are set to 0.001, 0.002, 0.01 and 2e-4, respectively. On Mini-Kinetics, we adopt a batch size of 48, and linearly scale the initial learning rates. The experiments with all patch sizes use the same aforementioned training configurations.

Sth-Sth V1&V2 and Jester (Section 4.2). When TSM [40] is implemented as the backbones in AdaFocusV2, the initial learning rates of f_{G}, f_{L}, f_{C} and π are set to 0.005, 0.01, 0.01 and 1e-4, respectively. The L2 regularization co-efficient is set to 5e-4. These changes follow the official implementation of TSM [40]. All other training settings are the same as the experiments on ActivityNet/FCVID in Section 4.1. All the experiments on Sth-Sth V1&V2 and Jester adopt the same training configurations.

[3]In most cases, we use the 224x224 ImageNet pre-trained models provided by PyTorch [50]. In AdaFocusV2-RN, since we deploy a ResNet-50 with down-sampled inputs (96^2) as f_{G}, we use the 96x96 ImageNet pre-trained ResNet-50 (provided by [66]).

C. Additional Results

Effectiveness of the learned patch selection policy is validated in Table 7. Following AdaFocusV1 [64], here we do not reuse the global feature $\boldsymbol{e}_t^{\mathrm{G}}$ for recognition for a clean comparison. We assume that our AdaFocusV2 network processes a fixed number of frames for all videos, and report the corresponding mAP on ActivityNet. Three pre-defined policies are considered as baselines: (1) randomly sampling patches, (2) cropping patches from the centres of the frames, and (3) sampling patches from a standard gaussian distribution centred at the frame. One can observe that the learned policies have considerably better performance, especially when only processing parts of all frames.

Visualization results are shown in Figure 10, where the green boxes indicate the locations of the image patches selected by AdaFocusV2-MN2/RN (96^2). It is observed that the model attends to the task-relevant regions of each frame, such as the dog, the dancer, the skateboard and the violin.