Supplementary Material for Look More but Care Less in Video Recognition

Anonymous Author(s)
Affiliation
Address
email

1 Experimental Settings

1.1 ImageNet

We first train our backbone network on ImageNet [2] using SGD optimizer. The L2 regularization coefficient and momentum are set to 0.0001 and 0.9 respectively. The learning rate decays by 0.1 at epochs 30 & 60. We train the network for 90 epochs with a batch size of 256 on 2 Nvidia Tesla V100 GPUs and adopt a 5-epoch warmup procedure.

1.2 Mini-Kinetics and ActivityNet

Then we add navigation module and train it along with the backbone network on video datasets. On Mini-Kinetics [8] and ActivityNet [1], we use SGD optimizer with a momentum of 0.9 and the L2 regularization coefficient is set to 0.0001. The initial learning rate is set to 0.002 and it will decay by 0.1 at epochs 20 & 40. The models are trained for 50 epochs with a batch size of 32 on 2 Nvidia Tesla V100 GPUs. The loss factor λ is set to 1 for the two datasets and the temperature τ decreases from 1 to 0.01 exponentially.

1.3 Jester and Something-Something

The training details on Jester [13] and Something-Something [4] datasets are the same with ActivityNet [1] except the following changes: the initial learning rate is 0.01 which decays at 25 & 45 epochs and it is trained for 55 epochs in total; the loss factor λ on these datasets is set to 0.5.

2 Building AFNet on BasicBlock

In previous experiments, we build AFNet on ResNet50 [5] which is made up of Bottleneck structure. Instead, we build AFNet with BasicBlock in this part on Jester dataset and compare it with baseline method TSN [17]. Table 1 shows that our method continuously shows significant advantages over TSN [17] in different selection ratios which demonstrate the effectiveness of our method on BasicBlock structure as well. Interestingly, AFNet obtains the best performance when the selection ratio is set to 0.5 and it shows relatively the lowest accuracy when selecting more frames. This can be explained that our navigation module effectively restrains the noise of meaningless frames and implements implicit temporal modeling which uses less frames but obtains higher accuracy.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Comparisons with baseline method on Jester.

Method	Frames	Jester Top-1 Acc.
TSN^{R34} [17]	8	83.5%
AFNet^{R34} (RT=0.75)	8	**89.3**%
AFNet^{R34} (RT=0.50)	8	**89.7**%
AFNet^{R34} (RT=0.25)	8	**89.5**%

3 Building AFNet on Other Backbones

In this part, we implement AFNet on stronger backbone, ResNet101 [5], and efficient backbone, MobileNet V3 [6] On Something-Something V1 to test its generalization ability.

Table 2: Comparisons with baseline method on Something-Something V1 with different backbones.

Method	Frames	Sth-Sth V1	
		Top-1 Acc.	GFLOPs
TSM^{R101} [11]	8	47.2%	62.8G
TSM^{R101} [11]	12	49.1%	94.2G
AFNet-TSM^{R101} (RT=0.4)	8	**47.2**%	**28.0G**
AFNet-TSM^{R101} (RT=0.4)	12	**49.8**%	**42.1G**
TSM^{MN3} [11]	8	42.2%	1.7G
TSM^{MN3} [11]	12	43.9%	2.6G
AFNet-TSM^{MN3} (RT=0.4)	8	**43.6**%	**1.5G**
AFNet-TSM^{MN3} (RT=0.4)	12	**45.3**%	**2.2G**

From Table 2, we can observe that AFNet continuously improves the accuracy of baseline methods while costing less computation. Specifically, when implementing on stronger backbone ResNet-101, AFNet significantly improves the efficiency by only costing 44.6% and 44.7% of the computations. When we build AFNet on efficient structure MobileNet V3, the saved computations are less obvious as the architecture of the base model is already very lightweight. Nevertheless, the improvement is more obvious compared to results on ResNet-101 which can be attributed to the effectiveness of the two-branch design and navigation module.

4 Building AFNet wtih More Frames

We build AFNet with more sampled frames in this section and compare it with baseline method. The results are shown in Table 3.

Table 3: Comparisons with baseline method on ActivityNet with more sampled frames.

Method	Frames	ActivityNet	
		mAP.	GFLOPs
TSN [17]	16	**76.9**%	62.8G
AR-Net [14]	16	73.8%	33.5G
VideoIQ [15]	16	74.8%	28.1G
AdaFocus [18]	16	75.0%	**26.6G**
AFNet (RS=0.4,RT=0.8)	16	76.6%	32.9G
TSN [17]	32	**78.0**%	131.2G
AFNet (RS=0.4,RT=0.8)	32	77.6%	**60.9G**

When sample 16 frames, TSN exhibits a clear advantage in performance over other efficient methods which can be explained by the information loss in the preprocessing phase (e.g., frame selection, patch cropping) of these dynamic methods. This phenomenon motivates us to design AFNet, which adopts a two-branch structure to prevent the loss of information. The results show that AFNet costs significantly less computation compared to baseline method with only a slight drop in performance. Furthermore, we conduct experiments on 32 frames and the phenomenon is similar to 16 frames.

5 Reflection on Implicit Temporal Modeling

In Section 3.2, we claim that the temporal masks result in learned frame-wise weights in each video which enforce implicit temporal modeling in AFNet. To better illustrate this point, we conduct the experiment by removing Gumbel softmax [7] in our navigation module and modifying it to learn soft temporal attention for the frames in focal branch. The result is shown in Table 4.

Table 4: Learning soft temporal weights on Something-Something V1.

Method	Frames	Sth-Sth V1 Top-1 Acc.
TSN [17]	8	18.6%
AFNet (RT=1.00)	8	**19.2**%
AFNet (RT=0.50)	8	**26.8**%
AFNet (RT=0.25)	8	**27.7**%
AFNet (soft-weight)	8	**27.0**%

We can make several conclusions from the table: (1) the learned weights significantly improve the performance of AFNet (RT=1.00) as it did not build any temporal modeling module like TSN, and the gain in performance can again demonstrate the effectiveness of our navigation module. (2) AFNet (soft-weight) has a similar performance to AFNet (RT=0.25) and AFNet (RT=0.50) which meets our expectations as we have analyzed in Section 3.2. Though the navigation module just learns a binary mask for each frame, it will decide whether the coefficient will be calculated for each frame at every convolutional block which results in learned temporal weights in each video. Learning a soft weight cause the same effect.

6 Computational Paradigms for Training and Inference

During training phase, the unselected frames at focal branch will be masked by zero values so that the feature will have the same shape with the residual feature and we can directly add them together. To avoid computation on the unimportant frames during inference, we will extract the salient frames from the original feature (slice operation on temporal dimension) before sending it to the convolutions. After that, we will create a zero tensor and rearrange the processed frames to the original temporal locations based on the learned mask. Note that temporal selection is made on batch dimension in real implementation so that it will not hurt the computational graph of vanilla convolutions.

7 Practical Speedup

We test the CPU (Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz) and GPU (NVIDIA GeForce GTX TITAN X) inference time of the competing methods. Note that all methods are implemented on ResNet-50 [5] and we use the original implementations provided by the authors. We sample 12 frames with the input size of 224x224 for all methods for fair comparison and get the average inference time over 100 runs in Table 5.

Table 5: GPU (NVIDIA GeForce GTX TITAN X) and CPU (Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz) inference time of competing methods with 12 sampled frames.

Method	GPU(ms)	CPU(ms)
bLVNet-TAM [3]	37	798
TANet [12]	27	595
SmallBig [9]	66	1268
TEA [10]	40	755
TSM [11]	**22**	633
AdaFocus-TSM [18]	45	744
AFNet-TSM (RT=0.4)	32	**422**

We can see from the table that AFNet achieves the fastest speed on CPU, making it favorable for employment on edge devices. While the GPU speed of AFNet is not as good as the speedup on CPU which shows inferior performance to static method TSM and TANet. The potential explanation is that the two-branch structure is less favorable in GPU acceleration and we do not have hardware-oriented optimization in our implementation yet. However, AFNet-TSM costs less inference time than dynamic method AdaFocus-TSM in both CPU and GPU.

8 More Ablation of AFNet

We further include more ablation of AFNet on ActivityNet with 12 sampled frames. First, we test AFNet's performance without the dynamic fusion module and the results in Table 6 can demonstrate that this design is nontrivial as it effectively balances the weights between the features from two branches. Besides, we explore different temperature decay schedules, including: 1) decay exponentially, 2) decay with a cosine shape, 3) decay linearly. The results show that exponential decay achieves the best performance and we adopt this as the default setting in all our experiments.

Table 6: Ablation of Fusion Strategy and Temperature Decay Schedule on ActivityNet.

Fusion Strategy	Temperature Decay Schedule	mAP RT=0.5
Dynamic Fusion	Exponential	**74.3**%
Addition	Exponential	73.5%
Dynamic Fusion	Cosine	74.1%
Dynamic Fusion	Linear	73.8%

9 Network Architecture

Here we take AFNet which builds on ResNet50 [5] as an example (denoted as AF-ResNet50) and we give detailed description of our network architecture in Table 7. Bottleneck is the residual block adopted in ResNet [5] which is made of 1×1, 3×3, 1×1 convolutions. Subscripts A and F stand for Ample Branch and Focal Branch respectively. A (Bottleneck, C) $\times N$ denotes N cascaded Bottlenecks with the channel size C. Note that we set the group number of convolution blocks in Focal branch to 2 in our network and we can build AFNet by substituting the first three stages of ResNet [5] with our proposed AF Module.

10 Limitations and Potential Negative Societal Impacts

First, the backbone of AFNet needs to be specially trained on ImageNet because of the two branch structure, while most other methods directly utilize the pretrained ResNet [5] from online resources. To make AFNet can be conveniently used by others, we offer the pretrained backbone on ImageNet

Table 7: Network Configuration of AFNet. Subscripts A and F stand for Ample branch and Focal branch respectively. A (Bottleneck, C) $\times$ N denotes N cascaded Bottlenecks with the channel size C. AFNet is built by replacing the first three stages of ResNet with AF Module.

Layers	Spatial output size	AF-ResNet50
Convolution	112×112	7×7, 64, s2
Pooling	56×56	MaxPooling
AF-Module	$(28 \times 28)_A$ $(56 \times 56)_F$ 28×28	(Bottleneck$_A$, 128) $\times$ 2 (Bottleneck$_F$, 256) $\times$ 2 Bottleneck, 256, s2
AF-Module	$(14 \times 14)_A$ $(28 \times 28)_F$ 14×14	(Bottleneck$_A$, 256) $\times$ 3 (Bottleneck$_F$, 512) $\times$ 3 Bottleneck, 512, s2
AF-Module	$(7 \times 7)_A$ $(14 \times 14)_F$ 14×14	(Bottleneck$_A$, 512) $\times$ 5 (Bottleneck$_F$, 1024) $\times$ 5 Bottleneck, 1024
ResBlock	7×7	(Bottleneck, 1024)$\times$3, s2
Pooling	1×1	7×7 Average Pooling
FC, Softmax	number of classes	

which can be accessed in our provided code. Second, we did not consider build any temporal modeling module during the design of AFNet which is the main focus of other static methods, like TEA [10], TDN [16], etc. However, we have demonstrated that AFNet implements implicit temporal modeling and it is compatible with existing temporal modeling module, like TSM [11]. To the best of our knowledge, our method has no potential negative societal impacts.

References

[1] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] Q. Fan, C.-F. Chen, H. Kuehne, M. Pistoia, and D. Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. *arXiv preprint arXiv:1912.00869*, 2019.

[4] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019.

[7] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[8] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[9] X. Li, Y. Wang, Z. Zhou, and Y. Qiao. Smallbignet: Integrating core and contextual views for video classification. In *CVPR*, 2020.

[10] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020.

[11] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

[12] Z. Liu, L. Wang, W. Wu, C. Qian, and T. Lu. Tam: Temporal adaptive module for video recognition. In *CVPR*, 2021.

[13] J. Materzynska, G. Berger, I. Bax, and R. Memisevic. The jester dataset: A large-scale video dataset of human gestures. In *ICCVW*, 2019.

[14] Y. Meng, C.-C. Lin, R. Panda, P. Sattigeri, L. Karlinsky, A. Oliva, K. Saenko, and R. Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *ECCV*, 2020.

[15] X. Sun, R. Panda, C.-F. R. Chen, A. Oliva, R. Feris, and K. Saenko. Dynamic network quantization for efficient video inference. In *ICCV*, 2021.

[16] L. Wang, Z. Tong, B. Ji, and G. Wu. Tdn: Temporal difference networks for efficient action recognition. In *CVPR*, 2021.

[17] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[18] Y. Wang, Z. Chen, H. Jiang, S. Song, Y. Han, and G. Huang. Adaptive focus for efficient video recognition. *arXiv preprint arXiv:2105.03245*, 2021.