Supplementary Material of "Delving into Details: Synopsis-to-Detail Networks for Video Recognition"

Shuxian Liang[1,2⋆], Xu Shen[2], Jianqiang Huang[2], and Xian-Sheng Hua[1⋆⋆]

[1] Zhejiang University
shuxian.lsx@zju.edu.cn, huaxiansheng@gmail.com,
[2] Alibaba Cloud Computing Ltd.
{shenxuustc,jianqiang.jqh}@gmail.com

In the supplementary material, we will provide more technical background, technical details, training details and efficiency analysis of our method. Specifically, we discuss the differences between S2DNet and some related works in Sec. 1. Then, we present more details about the detail sampler in Sec. 2 and more details about the training in Sec. 3. At last, we compare the efficiency of S2DNet to efficient action recognition methods in Sec. 4.

1 Differences between S2DNet and Some Related Works

Differences *vs.* Space-time Attention. The space-time attention is one of the basic components of the transformers-based works [1, 2, 4, 10]. The differences between the space-time attention and our space-time sampling lie in: 1) the attention uses global information and the sampling uses local information; 2) the attention works by importance weighting and the sampling works by Gaussian filtering or linear interpolation; 3) the attention is performed at the feature level while the sampling is performed at the input level (explained in Sec. 2).

Differences *vs.* Ensemble Models. Previous works for video recognition [5, 9, 12] use different variants of their proposed methods for ensemble, so as to further improve the accuracy. A key difference between the ensemble models and S2DNet is: the different variants for ensemble are parallel while SNet and DNet of S2DNet are arranged in a serial and progressive way. Notably, in our case, a direct ensemble of SNet and DNet is 4.0% worse on top-1 accuracy compared to the proposed S2DNet ($b1$ *vs.* $b0$ in Tab. 2b of the main paper).

2 Details about Detail Sampler

The representation of the top-k likely actions: hard top-k scheme *vs.* soft top-k scheme. Using hard top-k scheme (where s is a k-hot vector), DNet treats each of the k classes equally and is enforced to extract discriminative features that differentiate them. Using soft top-k scheme (where s is the softmax

⋆ This work was done when the author was visiting Alibaba as a research intern.
⋆⋆ Corresponding author.

predictions from SNet), DNet could take a shortcut by voting for actions with high scores in s. As a result, our hard top-k scheme ($a0$ in Tab. 2 of the paper) is +1.2% better on top-1 than the soft top-k scheme in our experiments.

Sampling at the Input Level. In our detail sampler, S2DNet adopts grid sampling over raw input frames while previous methods [8, 14] adopt grid sampling over features. The reason for input-level sampling is: it is too expensive for S2DNet to extract frame-wise features from the input frames as dense as $\mathcal{I}$.

While applying the grid sampling at the input level, a new challenge emerges: if the key frames/regions are missed by the input-level sampling, it is difficult to recover them at later stages[3]. This leads to a high variance of the gradients received by the location & scale module f_θ, hindering the optimization process.

To mitigate the problem, firstly, the gradients of the output volume $\hat{\mathcal{V}}$ are normalized to $(-1, 1)$ before they are back-propagated to the module f_θ. Secondly, using Kaiming initializer [7] with $a = 5$, we initialize (μ_t, μ_x, μ_y) around 0, (δ_x, δ_y) around 1, and δ_t around $\frac{T}{2\times T_d}$. These settings help stabilize the gradients and enable the detail sampler to start from an appropriate initialization at the beginning of training.

Scaling Location & Scale Parameters. The location & scale parameters θ are scaled before they are fed to the detail sampler. Specifically, the grid centers (μ_t, μ_x, μ_y) are scaled to $(-1, 1)$ by:

$$\begin{aligned} \hat{\mu_t} &= \tanh(\mu_t), \\ \hat{\mu_x} &= \tanh(\mu_x), \\ \hat{\mu_y} &= \tanh(\mu_y), \end{aligned} \tag{1}$$

which ensures the grid centers are within the sampling input. The grid strides $(\delta_t, \delta_x, \delta_y)$ are processed by:

$$\begin{aligned} \hat{\delta_t} &= d_t \exp(\delta_t), \\ \hat{\delta_x} &= d_x \exp(\delta_x), \\ \hat{\delta_y} &= d_y \exp(\delta_y), \end{aligned} \tag{2}$$

where exp is used to ensure positivity, and $d_t = \frac{T}{2\times T_d}$, $d_x = 1$ and $d_y = 1$ refer to the aforementioned initial values of these strides.

Choices of Filters. To mimic the foveation of human eyes, early works [6, 14] uses Gaussian filters for spatial sampling. [8] shows that the mixture of a temporal linear filter (based on linear interpolation) and two spatial Gaussian filters work well for spatio-temporal sampling. For S2DNet, as shown in Tab. 1, given the dense input $\mathcal{I}$, using Gaussian filters for both sampling achieves comparable results (-0.1%) while being significantly faster than the mixture ($\downarrow 5.9\times$). Hardware-oriented optimization might speed up the temporal linear filter but that is beyond the scope of this paper. Thus, we use Gaussian filters for both spatial and temporal sampling for S2DNet.

[3] By contrast, for feature-level sampling, the features are able to keep the information about the neighborhood and thus make it possible to recover the missing frames/regions.

Table 1: Comparison of the filters for the detail sampler on Something-Something V2. Latency denotes the inference time (ms) per video using a Tesla V100 with batch size as 16. Code is implemented using PyTorch [11]

Temporal	Spatial	Latency (ms)	top-1
Linear	Gaussian	38.2	**62.6**
Gaussian	Gaussian	**6.5**	62.5

Configurations of fusing contextual features m to detailed features. As shown in Table 2, compared to our default settings ($d0$), using m in detail backbone/classifier alone ($d1$/$d2$) degrades the top-1 result by 1.4%/0.6%. Moreover, fusing m after *res2* & *res3* ($d0$) has +0.5%/+0.2% better performance than those involving less or more stages ($d3$/$d4$). The above results suggest that the fusion of m is more effective on high-level detail features.

Table 2: Results of different configurations of fusing contextual features m to detailed features. "cls." denotes the detail classifier

	res2	res3	res4	cls.	top-1	top-5	GFLOPs
$d0$		✓	✓	✓	**74.7**	**91.1**	18.0
$d1$		✓	✓		74.1	90.7	18.0
$d2$				✓	73.3	90.9	17.9
$d3$			✓	✓	74.2	90.9	18.0
$d4$	✓	✓	✓	✓	74.5	91.1	18.1

3 Training Details

Training Algorithm. Since all components of S2DNet are differentiable, our network can be trained from scratch in an end-to-end manner. In practice, for a faster convergence, we adopt a two-stage training algorithm as follows. In the first stage, the location & scale parameters θ are frozen. The center crop of uniform input frames is used as input for DNet and all components except the the location & scale module f_θ are trained jointly. In the second stage, the parameters θ are activated and all components of S2DNet are trained in an end-to-end manner.

Training Hyper-parameters. For all datasets, S2DNet is trained using a SGD optimizer with a momentum of 0.9. For Mini-Kinetics and Something-Something V1 & V2, instantiated using TSM, the backbones are initialized from ImageNet [3] pretrained weights. The training parameters are: 120 training epochs (40 for the first stage and 80 for the second), initial learning rate 0.001

(with a cosine learning rate decaying schedule), weight decay 0.001, batch size 64 and dropout rate 0.5. For Kinetics-400, instantiated using X3D, the backbones are initialized randomly. The training parameters are: 150 training epochs (50 for the first stage and 100 for the second), initial learning rate 0.005 (with a cosine learning rate decaying schedule), weight decay 0.001, batch size 64 and dropout rate 0.5.

4 More Efficiency Analysis

This subsection provides the efficiency analysis of the S2DNet variant in Tab. 3b of the main paper. Notably, this variant is compared to the efficient action recognition methods (*e.g.*, [13]) while the one in Sec. 4.5 of the main paper is compared to the state-of-the-art methods (*e.g.*, [12]). For all methods in comparison, the testing batch size is set as 16 and a single Tesla V100 is used. As shown in Tab. 3, one can observe that S2DNet is more accurate and enjoys significant practical speedup with minor extra parameters. With comparable backbones and similar/more frames, S2DNet achieves its efficiency gain by reducing redundant computation for spatial processing. More sophisticated implementation (*e.g.*, hardware-oriented optimization) might bring in more efficiency gain. This is left to our future work.

Table 3: Efficiency comparison on Something-Something V2. All measures use a Tesla V100 with batch size as 16. † denotes the reproduced results using the corresponding official code

Methods	Backbones	Frames	Spatial Size	GFLOPs	Param.	Latency (ms)	top-1
TSM [9]	R50	8	224^2	33	**24.3M**	8.4	59.1
AdaFocus [13]	MN2/R50	8/12	$224^2/176^2$	34	26.3M†	16.0†	60.7
S2DNet (ours)	MN2/R50	8/12	$144^2/144^2$	**22**	28.0M	**6.5**	**62.5**

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. arXiv preprint arXiv:2103.15691 (2021)
2. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? arXiv preprint arXiv:2102.05095 (2021)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
4. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. arXiv preprint arXiv:2104.11227 (2021)
5. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV (2019)
6. Gregor, K., Danihelka, I., Graves, A., Rezende, D., Wierstra, D.: Draw: A recurrent neural network for image generation. In: ICML (2015)
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV (2015)
8. Huang, Z., Xue, D., Shen, X., Tian, X., Li, H., Huang, J., Hua, X.S.: 3d local convolutional neural networks for gait recognition. In: ICCV (2021)
9. Lin, J., Gan, C., Han, S.: Tsm: Temporal shift module for efficient video understanding. In: ICCV (2019)
10. Neimark, D., Bar, O., Zohar, M., Asselmann, D.: Video transformer network. arXiv preprint arXiv:2102.00719 (2021)
11. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS **32** (2019)
12. Wang, L., Tong, Z., Ji, B., Wu, G.: Tdn: Temporal difference networks for efficient action recognition. In: CVPR (2021)
13. Wang, Y., Chen, Z., Jiang, H., Song, S., Han, Y., Huang, G.: Adaptive focus for efficient video recognition. In: ICCV (2021)
14. Yang, J., Shen, X., Tian, X., Li, H., Huang, J., Hua, X.S.: Local convolutional neural networks for person re-identification. In: ACM MM (2018)