Supplementary Material: Supervoxel Attention Graphs for Long-Range Video Modeling

Yang Wang[1], Gedas Bertasius[2], Tae-Hyun Oh[3], Abhinav Gupta[2], Minh Hoai[1], Lorenzo Torresani[2]
[1]Stony Brook University, [2]Facebook AI, [3]POSTECH
{wang33,minhhoai}@cs.stonybrook.edu,taehyun@postech.ac.kr,{gedas,gabhinav,torresani}@fb.com

1. Experimental Evaluation on Something^2-V2

While our main submission includes a comprehensive evaluation on Charades and Something^2-V1, here we present results obtained on a third dataset, Something^2-V2 [5]. The Something^2-V1 and V2 datasets have the same 174 fine-grained action categories, such as "*Putting something on a surface*", "*Putting something into something*" *etc*. The V2 dataset has a total number of 220,847 videos, which is more than twice as many videos as V1. For this evaluation, we leverage as backbone the same R101 network utilized in TSM [8]. We refer the reader to Sec. 5.2 of this Supplementary Material for further information on the experimental setup and the implementation details. The results are reported in Table 1. Compared to the TSM baseline, our proposed Supervoxel Attention Graph (SVAG) applied to the supervoxel similarity graph $\mathbf{G}^{\texttt{sim}}$, improves the top-1 accuracy by 1% (from 62.9% to 63.9%).

We point out that SVAG provides consistent gains over the baselines across all datasets (Charades, Something^2-V1, and Something^2-V2) and all backbones (R50, R101, and ip-CSN152) considered in our experiments.

2. Experimental Evaluation on Kinetics-400

We also perform experiments on Kinetics-400 [1], another standard benchmark for action recognition in videos. It contains about 260K videos of 400 different human action categories. We use the training split (240K videos) for training and the validation split (20K videos) for evaluation. For this evaluation, we leverage as backbone the same ip-CSN152 network utilized in CSN [10]. We refer the readers to Sec. 5.3 of this Supplementary Material for further information on the experimental setup and the implementation details. The results are reported in Table 2. Compared to the CSN baseline, our proposed Supervoxel Attention Graph (SVAG) applied to the supervoxel similarity graph $\mathbf{G}^{\texttt{sim}}$, improves the top-1 accuracy by 0.5% (from 80.3% to 80.8%).

3. Comparison to Spatiotemporal Graphs

Wang and Gupta [11] suggested another type of hand-designed graph, called spatiotemporal graph, in addition to the similarity graph. While their graphs leverage region proposals as nodes, here we discuss how we can construct spatiotemporal graphs on our supervoxels. We then present an empirical evaluation demonstrating that our learnable similarity graph on supervoxels is superior to the spatiotemporal graph on supervoxels and that the former effectively already encodes the information represented in the latter.

Spatiotemporal Graphs. In order to adapt the spatiotemporal graph of Wang and Gupta [11] to operate on supervoxels, we need to define the notion of spatial and temporal proximity among supervoxels, which have irregular shapes. We achieve this goal by measuring the amount of contact between two adjacent supervoxels along the different dimensions (spatial and temporal).

For the *temporal graph*, we build two graphs: a forward graph $\mathbf{G}^{\texttt{front}}$ and a backward graph $\mathbf{G}^{\texttt{back}}$. Given supervoxels i and j, we set edge $\mathbf{G}^{\texttt{front}}_{i \leftarrow j}$ by counting the number of pixels of j that become part of supervoxel i when we move forward by one frame (i.e., such that pixel (x, y, t) belongs to j but $(x, y, t+1)$ belongs to i). As in [11], we normalize the edge weight so that the sum of the edge values connected to supervoxel i will be 1. Edge $\mathbf{G}^{\texttt{back}}_{i \leftarrow j}$ is computed similarly but moving backward by one frame.

For the *spatial graph* $\mathbf{G}^{\texttt{spatial}}$, we compute the adjacency between supervoxels along the two spatial axes in a similar way to the the temporal graph. In this case $\mathbf{G}^{\texttt{spatial}}_{i \leftarrow j}$ is set by counting how many pixels (x, y, t) of j become part of i when moving by 1 along either the x-axis (i.e., to $(x+1, y, t)$ or $(x-1, y, t)$) or the y axis (i.e., to $(x, y+1, t)$ or $(x, y-1, t)$). Again, we normalize this count so that the sum of the edge values connected to supervoxel i will be 1.

Spatiotemporal Graph Convolution. We define the complete spatiotemporal graph as $\mathbf{G}^{\texttt{s-t}} = (\mathbf{G}^{\texttt{spatial}}, \mathbf{G}^{\texttt{front}}, \mathbf{G}^{\texttt{back}})$. As in [11], due to the usage of multi-graphs in $\mathbf{G}^{\texttt{s-t}}$, we extend the definition the graph

Method	Backbone	Pretrain	# Frame × Crop	Top-1 val.	Top-5 val.	Top-1 test	Top-5 test
TSM [8]	R101	Kinetics-400	16 × 1	62.9	88.1	–	–
SVAG (Ours)	R101	Kinetics-400	16 × 1	**63.9**	88.6	62.4	88.3

Table 1: **Comparison between TSM [8] and SVAG on Something^2-V2**. We use the same backbone (R101) and the same pretraining scheme. SVAG provides a gain of 1% in top-1 accuracy over TSM.

Method	Backbone	Pretrain	# Frame × Crop	Top-1 val.	Top-5 val.
CSN [10]	ip-CSN152	IG-65M	32 × 10	80.3†	94.6
SVAG (Ours)	ip-CSN152	IG-65M	32 × 10	**80.8**	94.7

Table 2: **Comparison between CSN [10] and SVAG on Kinetics-400**. We use the same backbone (ip-CSN152) and the same pretraining scheme. SVAG provides a gain of 0.5% in top-1 accuracy over CSN. † indicates the performance of our own implementation with 10-crop testing (80.8%) while the original paper [10] used 30-crop testing (82.5%). Our conclusion holds due to consistent relative improvement.

convolutional layer with multiple graphs ($\mathbf{G}^i$) as:

$$\texttt{GConv}[(\mathbf{G}^i), \mathbf{Z}] \coloneqq \textstyle\sum_i \texttt{LN}[\mathbf{G}^i f_i(\mathbf{Z})\mathbf{W}_{r,i}], \qquad (1)$$

Comparing $\mathbf{G}^{\texttt{sim}}$ and $\mathbf{G}^{\texttt{s-t}}$. Table 3 provides a comparison of our SVAG model applied to $\mathbf{G}^{\texttt{sim}}$ only, $\mathbf{G}^{\texttt{s-t}}$ only, or a combination of the two. The training and testing are done on the Charades and Something^2 datasets.

First, from the results on the Charades dataset, it can be seen that SVAG applied to $\mathbf{G}^{\texttt{sim}}$ provides superior accuracy compared to when applied to $\mathbf{G}^{\texttt{s-t}}$ (44.1 vs 42.8 in mAP). Second, applying SVAG to the combined $\mathbf{G}^{\texttt{sim}}+\mathbf{G}^{\texttt{s-t}}$ does not yield further improvement over using $\mathbf{G}^{\texttt{sim}}$ only. A similar trend is also hold in the results on the Something^2 dataset. This suggests that the learned similarity in $\mathbf{G}^{\texttt{sim}}$ already captures most of the spatiotemporal information encoded in $\mathbf{G}^{\texttt{s-t}}$.

4. Visualization of Supervoxel Attentions

In Fig. 1 we provide more video examples that visualize the supervoxel attentions. We observe the highly attended supervoxels are the most relevant to the actions performed in the video. This demonstrates our model spontaneously learns to attend to human-centric supervoxels and objects likely being interacted even without any spatial annotation.

5. Implementation Details

5.1. Experiments on Charades

Backbone Model (R101-NL). We use the "R101-NL" model provided by [12] as our backbone model to extract video features on Charades dataset [9]. Following Wu *et al*., the R101-NL model is constructed as follows. The backbone model is based on the 2D ResNet-101 [6] architecture with non-local operations [11]. The model was first pre-trained on the ImageNet dataset [3], and subsequently inflated into a 3D CNN using the I3D technique [2], *i.e*., inflating 2D $k\times k$ kernels into 3D $t\times k\times k$ kernels by copying the 2D kernel weights t times and rescaling by $1/t$. After inflation, the 3D CNN was trained on Kinetics-400 [2] and then finetuned on the target dataset, *i.e*., Charades. We remove the last classification layer and use the fully convolutional part of the network for feature extraction. The backbone model "R101-NL" takes as input a sequence of RGB frames of shape $T\times H\times W\times 3$, and outputs feature maps of shape $\frac{T}{2}\times\frac{H}{16}\times\frac{W}{16}\times 2048$.

Input Preprocessing. Since videos have different aspect ratios and sizes, we process them to have the same spatiotemporal dimensions. We follow the same procedure used in [12]. Given an input video, we first reduce its temporal resolution to 6 frames per second. We then spatially resize the videos such that the shorter side has 256 pixels while the aspect ratio is kept the same, and crop them with a spatial window of size 256×256 at each corner and center location. For instance, given an input video clip of 10 seconds, the input to the backbone model is of shape $60\times256\times256\times3$, and the output feature maps $\mathbf{V}$ is of shape $30\times16\times16\times2048$.

Hyper-Parameters. For optimization on Charades, we use SGD (with momentum set to 0.9) and a weight decay of 1.25×10^{-5}. The learning rate starts with 0.02 and decays with a rate of 0.1 at the 10-th epoch. The training stops after 15 epochs. The training batch size is set to 16.

Dataset	Charades			Something^2-V2		
Model	Backbone	Pretrain	mAP	Backbone	Pretrain	Top-1 val.
SVAG w/ $\mathbf{G}^{\mathtt{s-t}}$	R101-NL	Kinetics-400	42.8	R101	Kinetics-400	63.6
SVAG w/ $\mathbf{G}^{\mathtt{sim}}$	R101-NL	Kinetics-400	**44.1**	R101	Kinetics-400	**63.9**
SVAG w/ $\mathbf{G}^{\mathtt{sim}}$ + $\mathbf{G}^{\mathtt{s-t}}$	R101-NL	Kinetics-400	**44.1**	R101	Kinetics-400	63.7

Table 3: **Comparison between SVAG applied to the similarity graph $\mathbf{G}^{\mathtt{sim}}$ vs. the spatiotemporal graph $\mathbf{G}^{\mathtt{s-t}}$ on Charades (left) and Something^2-V2 (right)**. SVAG applied to $\mathbf{G}^{\mathtt{sim}}$ yields consistently better results than when used on $\mathbf{G}^{\mathtt{s-t}}$. Furthermore, SVAG applied to a combination of $\mathbf{G}^{\mathtt{sim}}$ and $\mathbf{G}^{\mathtt{s-t}}$ does not improve over inference on $\mathbf{G}^{\mathtt{sim}}$ only. This suggests that the learnable similarity encoded in $\mathbf{G}^{\mathtt{sim}}$ already captures most of the information contained in $\mathbf{G}^{\mathtt{s-t}}$. We report mAP and top-1 validation accuracy for Charades and Something^2-V2, respectively.

For supervoxel computation, we use the following parameters:

$$(\lambda_{RGB}, \lambda_{semantic}, \lambda_T, \lambda_{XY}) = \left(5, 2, \tfrac{K_t}{0.1T}, \max(\tfrac{K_h}{0.1H}, \tfrac{K_w}{0.1W})\right), \quad (2)$$

where K_w, K_t, and K_h are the number of supervoxels along the (x, y, t)-axes, respectively, RGB and segmentation channels are in range $[0, 1]$. We refer the readers to [7] for more information on the setting of these hyperparameters.

5.2. Experiments on Something^2 (V1 and V2)

Backbone Model (TSM). TSM [8] stands for Temporal Shift Module. It uses a 2D ResNet-101 to extract video features, and shifts part of the features along the temporal dimension to achieve temporal modeling at zero computation and zero parameters. TSM takes as input a sequence of RGB frames of shape $T\times H\times W\times 3$, and outputs feature maps of shape $T\times\frac{H}{16}\times\frac{W}{16}\times 2048$.

Input Preprocessing for TSM. To prepare the input RGB frames for the TSM model, we evenly sample 16 frames from each video and spatially resize them such that the shorter side has 256 pixels while the aspect ratio is kept the same. We then crop them with a spatial window of size 256×256 at the center location. The input video to the TSM model has shape $16 \times 256 \times 256 \times 3$, and the final representation is of size $16 \times 16 \times 16 \times 2048$.

Backbone Model (CSN). CSN [10] refers to the Channel-Separated Convolutional Network. Compared to standard 3D CNNs, CSN achieves better accuracy and lower computational cost by factorizing 3D convolutions via separating channel interactions and spatiotemporal interactions. We refer the readers to [10] for mode details. Specifically, we use "ip-CSN152" as our backbone model to extract video features. The model was first pre-trained on the IG-65M [4] dataset and then finetuned on the target dataset, *i.e.*, Something^2-V1. The ip-CSN152 model takes as input a sequence of RGB frames of shape $T\times H\times W\times 3$, and outputs feature maps of shape $\frac{T}{8}\times\frac{H}{32}\times\frac{W}{32}\times 2048$.

Input Preprocessing for CSN. To process the Something^2 videos as input to the CSN model, we first change their temporal resolution to 15 frames per second, and spatially resize them such that the shorter side has 256 pixels while the aspect ratio is kept the same. We then crop the videos with a spatial window of size 256×256 at the center location. After these steps, we evenly sample five time-locations from each video. At each of the five locations, we take a 32-frame sequence and feed it to the CSN backbone. The output feature maps at each location is of shape $4 \times 8 \times 8 \times 2048$, which is subsequently average-pooled along the temporal dimension to have shape $1 \times 8 \times 8 \times 2048$. Finally, we temporally concatenate the feature maps from all five locations as the final feature representation for each video, which is of shape $5 \times 8 \times 8 \times 2048$.

Hyper-Parameters The Something^2 dataset has two versions, *i.e.*, V1 and V2. The hyper-parameters here are the same as for the Charades experiments, except that the learning rate starts with 0.001 and decays at the end of epoch 4. The training stops after 6 epochs for V1. For V2, the learning rate decays at the end of epoch 2, and the training stops after 4 epochs. The training takes fewer epochs on V2 because V2 includes roughly twice the amount of training data compared to V1. The training batch size is set to 64 for both V1 and V2.

5.3. Experiments on Kinetics-400

For Kinetics-400, we also leverage as backbone the ip-CSN152 network proposed in [10]. The implementation details and the hyper-parameter setup of the experiments on Kinetics-400 are the same with those of the experiments on Something^2 V2.

Figure 1: **Visualization of the attention learned by our proposed Supervoxel Attention Graph (SVAG).**

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017.

[2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017.

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

[4] D. Ghadiyaram, D. Tran, and D. Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proc. CVPR*, 2019.

[5] R. Goyal, S. E. 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 *Proc. ICCV*, 2017.

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

[7] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz. Superpixel sampling networks. In *Proc. ECCV*, 2018.

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

[9] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proc. ECCV*, 2016.

[10] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 2019.

[11] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proc. CVPR*, 2018.

[12] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick. Long-term feature banks for detailed video understanding. In *Proc. CVPR*, 2019.