Supplementary Material for CVPR 2020 paper Something-Else: Compositional Action Recognition with Spatial-Temporal Interaction Networks

Joanna Materzynska
University of Oxford, TwentyBN

Tete Xiao
UC Berkeley

Roei Herzig
Tel Aviv University

Huijuan Xu*
UC Berkeley

Xiaolong Wang*
UC Berkeley

Trevor Darrell*
UC Berkeley

1. Object appearance features

We additionally explore the effectiveness of our method when applied to features corresponding to the object descriptors. To this end, we follow [11] to extract the object appearance features. We apply our STIN model replacing the bounding box features with the convolutional ones. We test this in the few-shot compositional setting. We use a benchmark model that combines STIN + OIE + NL using bounding box features and object appearance feature, noted as STIN + OIE + NL [BB + OA] as well as model that combines appearance feature from the I3D network and above mentioned model trained jointly; I3D + STIN + OIE + NL [BB + OA] and ensemble I3D, STIN + OIE + NL [BB + OA]. We present the results in Table 1.

	base		*few-shot*	
model	top-1	top-5	5-shot	10-shot
STIN	54.0	78.9	14.2	19.0
STIN + OIE	58.2	82.6	16.3	20.8
STIN + OIE + NL	58.2	82.6	17.7	20.7
I3D	73.6	92.2	21.8	26.7
I3D + STIN + OIE + NL	76.8	93.3	23.7	27.0
I3D, STIN + OIE + NL	76.1	92.7	27.3	32.6
STIN + OIE + NL [BB + OA]	71.6	93.0	31.7	37.2
I3D + STIN + OIE + NL [BB + OA]	73.5	97.3	31.9	36.6
I3D, STIN + OIE + NL [BB + OA]	80.9	96.1	36.8	43.1

Table 1. **Few-shot compositional action recognition** on *base* categories and *few-shot novel* categories. The results shown are using detection boxes.

The above experiments suggest that our STIN model can also generalize to different types of features. Combining boxes features with object appearance features gives 17.7 % improvement on the *base* validation set, 17.5% and 18.2 % improvement on the few-shot setting. Combining this model with I3D also significantly boosts performance on all the validation sets.

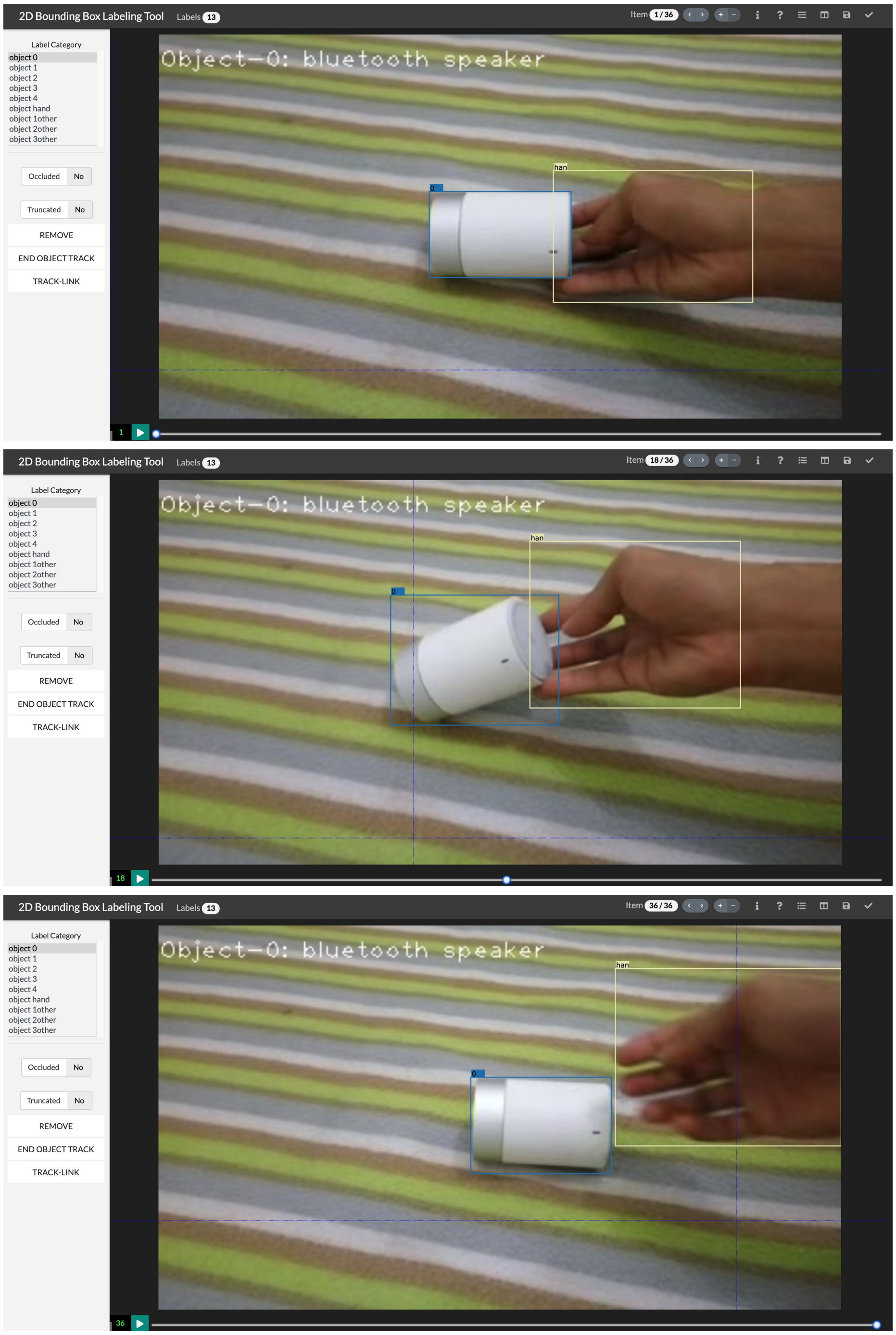
2. Annotation Details

We provide bounding box annotations for hands and objects parsed from the video captions on the **Something-Something V2** dataset [3] and perform our experiments on the annotated dataset. We convert the videos into frames using 12 fps. Though there are hundreds of frames for each video, the annotation tool that we use has built-in tracking function and saves us much time. The annotators only need to selectively annotate several key frames in the videos and the tool will interpolate the bounding boxes in the rest of frames perfectly. The total number of annotated videos are 180,049. The average time taken

*Equal advising

for a single video is 1.7 minutes, and the total time taken for the whole annotation project is approximately 5101 hours. In page 3, we provide the visualization of the annotation interface that we use in our project.

Visualization of the annotation interface for video object bounding box annotation



Throw bluetooth speaker

3. Object Detector and Tracker Details

We illustrate more details of the implementation of the detector and the tracker as below.

Detector. We choose Faster R-CNN [10] with the Feature Pyramid Network (FPN) [8] and ResNet-101 [5] backbone. RoI-Pooling in the original implementation is replaced by RoI-Align [4]. We use an open-sourced implementation [12] and closely follow default training settings, except for a handful of difference: We set starting learning-rate as 0.02, and train the model on 10 GPUs, with each GPU hosting 4 images. For training, images are resized such that their shorter edge is randomly selected among $[512, 544, 576, 608, 640]$ pixels, whereas for testing it is $[400, 500, 576, 600]$ pixels. During testing we also flip the image for ensemble. The model is initialized by weights pre-trained on COCO [9] dataset, which has 42.0 detection mAP on COCO `val` subset. Only two categories are registered for the detector: *hand* and *object*. Fine-tuning takes about 2.5 days on 10 GPUs. We take detections whose scores are greater than 0.65.

Tracker. We use Kalman Filter [6] and Kuhn-Munkres (KM) algorithm [7] for the tracking components, as in [1]. At each time step, the Kalman Filter predicts plausible whereabouts of instances in current frame based on previous tracks, then the predictions are matched with single-frame detections by KM algorithm. The matching weight of two bounding-boxes, denoted as $u = (x_\mathrm{u}, y_\mathrm{u}, h_\mathrm{u}, w_\mathrm{u})$ and $v = (x_\mathrm{v}, y_\mathrm{v}, h_\mathrm{v}, w_\mathrm{v})$ representing the center, height and width of the box, are defined by a weight function $g(u, v)$. Matched predictions are updated by detections in the current frame. To partially deal with instance occlusion and re-entering, unmatched predictions are kept for additional T_k frames until they are discarded; unmatched detections are registered as a possible new instance, if their presence continue for at least T_c frames. We keep at most top-k tracklets in each video, sorted by the sum of detection scores of individual instance in each frame averaged by the length of the tracklet (score of unmatched prediction is set as 0). Finally, we filter out the tracklets whose scores are below S. Note that no appearance feature is used in the tracking framework.

In our implementation, we set $T_k = 6$, $T_c = 8$, $k = 4$ and $S = 0.7$. Since the instances (both subjects and objects) can move relatively fast in human-object interactions, instead of using Intersection-over-Union (IoU) as matching function g, we use “regression energy“ like in [2]:

$$
\begin{aligned}
g(u,v) = &|\frac{x_\mathrm{v} - x_\mathrm{u}}{w_\mathrm{u}}| + |\frac{y_\mathrm{v} - y_\mathrm{u}}{h_\mathrm{u}}| \\
&+ |\log \frac{w_\mathrm{v}}{w_\mathrm{u}}| + |\log \frac{h_\mathrm{v}}{h_\mathrm{u}}|
\end{aligned} \tag{1}
$$

The maximum regressable energy is set as 6, and instances from different classes (hand and object) have infinite regression energy.

References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468. IEEE, 2016. 3

[2] Ross Girshick. Fast r-cnn. In *CVPR*, pages 1440–1448, 2015. 3

[3] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The” something something” video database for learning and evaluating visual common sense. In *ICCV*, 2017. 1

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[6] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 3

[7] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 3

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 3

[11] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, pages 399–417, 2018. 1

[12] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 3