Is an Object-Centric Video Representation Beneficial for Transfer?

Supplementary Material

Chuhan Zhang[1], Ankush Gupta[2], and Andrew Zisserman[1]

[1] Visual Geometry Group, Department of Engineering Science
University of Oxford
{czhang,az}@robots.ox.ac.uk
[2] DeepMind, London
ankushgupta@google.com

Table of Contents

1 Architecture

1.1 Visual backbone

We use Motionformer [1] as the visual backbone, it takes a sequence of video frames $I \in \mathbb{R}^{T\times H\times W\times 3}$, patchifies these into 3D patches of size $(2\times 16\times 16\times 3)$ each, and then encodes them. It has 12 self-attention layer with 12 heads each, and outputs feature maps $V \in \mathbb{R}^{T\times (H'W')\times C}$, where $C = 768$. All the hyper-parameters are the same as the ones used in [1] on SomethingSomething-V2.

1.2 Trajectory backbone

The trajectory backbone consists of a box embedding module and a Spatial Temporal Layout model (STLT) of [2]. It takes a sequence of bounding boxes of objects as input and outputs spatial and temporal layout embeddings.

The input bounding boxes $B^t = (b_1^t, b_2^t, b_3^t, \ldots, b_o^t)$ of $\mathcal{O}$ number of objects in a given frame are in the format $[x_1, y_1, x_2, y_2]$, they are first projected into box embeddings $\Phi \in \mathbb{R}^{O\times T\times C}$ through an MLP. Learnable object-ID embeddings $\mathcal{D} = \{d_j\}_{j=1}^{O}$ are added to the box embeddings to obtain Φ_{in}, which serve as an input to the STLT .

STLT consists of two self-attention transformers, the Spatial Transformer and the Temporal Transformer. An overview of its archiecture is shown in Figure 1. The Spatial Transformer processes the boxes at each frame separately. In each frame, it takes a learnable `CLS` token and box embeddings $\Phi_{in}^t \in \mathbb{R}^{O\times C}$ as input into the self-attention layers, and output a frame-level representation $l^t \in \mathbb{R}^{1\times C}$ and spatial-context-aware box embeddings $\Phi_{out}^t \in \mathbb{R}^{O\times C}$.

The Temporal Transformer encodes trajectory information between frames, it applys self-attenion on the frame-level embeddings $L^t = (l^1, l^2, \ldots, l^T)$ from the Spatial Transformer with another learnable `CLS` token. At the output, we will have temporal-context-aware frame embeddings $L_{out} \in \mathbb{R}^{T\times C}$ and a video-level representation $C_{traj} \in \mathbb{R}^{1\times C}$. C_{traj} is later used to compute the classification loss, while $L_{out} \in \mathbb{R}^{T\times 1\times C}$ is concatenated with the $\Phi_{out} \in \mathbb{R}^{T\times O\times C}$ from the Spatial Transformer as the final trajectory embeddings $G \in \mathbb{R}^{T\times (O+1)\times C}$, G is then broadcasted and concatenated with the visual feature map $V \in \mathbb{R}^{T\times H'W'\times C}$ from the 6th layer of the visual backbone, to be the keys and values of the Object Learner.

1.3 Object Learner

The Object Learner consists of 4 cross-attention layers, each with 4 heads. The input feature dimension is 512 and the feed-forward dimension is 2048. Keys, values and the object-ID embeddings $\mathcal{D}$ from the backbones are linearly projected to dimension 512. Queries into the Object Learner are the sum of the projected object-ID embeddings $\mathcal{D}' = \{d_i'\}_{i=1}^{O}$ and a set of learnable embeddings $\mathcal{Q} = \{q_i\}_{i=1}^{O+K}$, where O is the number of object queries and K is the number of context queries. We set $O = 5$, $K = 3$ in the experiments.

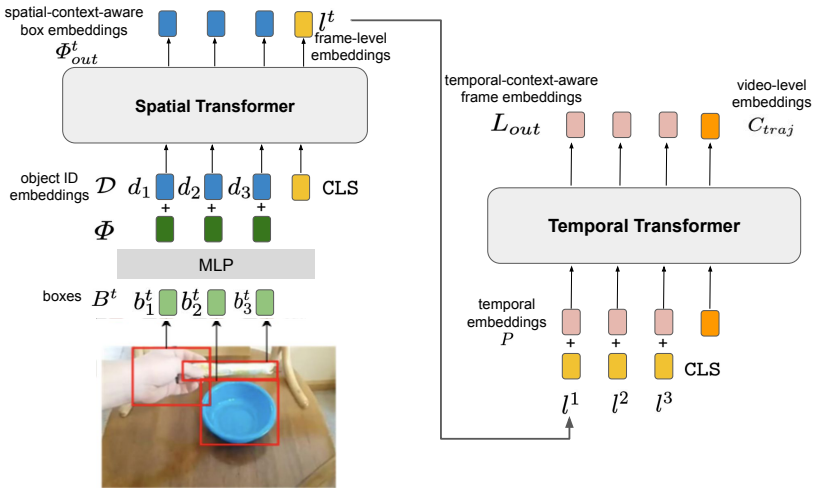

Fig. 1. Architecture of STLT. Image reproduced from [2]. **Left: Spatial Transformer:** It takes object bounding boxes at different frames and encodes their spatial layout independently, a special class embedding is used to aggregate per-frame information. **Right: Temporal Transformer:** It takes the frame-level output from the spatial Transformer as input, and encodes them along the temporal dimension. Another special class embedding is concatenated to the temporal features for video-level classification.

We use trajectory attention mechanism [1] in cross-attention layers to replace joint spatio-temporal attention. For each object query q_i, we first compute its attention scores along the spatio-temporal dimension, and use the scores for weighted pooling on spatial dimension only:

$$\mathbf{a}_{ist} = \frac{\exp\langle \mathbf{q}_i, \mathbf{k}_{st} \rangle}{\sum_{s't'} \exp\langle \mathbf{q}_i, \mathbf{k}_{s't'} \rangle}, \tag{1}$$

$$\tilde{\mathbf{y}}_{it} = \sum_s \mathbf{v}_{st} \cdot \mathbf{a}_{ist}, \tag{2}$$

where $\tilde{y}_{it}$ is the spatially aggregated token at time t given q_i, which is also referred as the 'trajectory token' at time t. Once the trajectories $\tilde{Y}_i$ are computed, they are further pooled across time to extract intra-frame information/connections. To do so, the trajectory tokens are projected to a new set of keys and values, and the query is projected again to a new set of temporal queries:

$$\tilde{\mathbf{q}}_i = \tilde{\mathbf{W}}_q \, \mathbf{q}_i, \quad \tilde{\mathbf{k}}_{it} = \tilde{\mathbf{W}}_k \, \tilde{\mathbf{y}}_{it}, \quad \tilde{\mathbf{v}}_{it} = \tilde{\mathbf{W}}_v \, \tilde{\mathbf{y}}_{it}. \tag{3}$$

The new query is used to pool across the new time (trajectory) dimension by applying 1D cross-attention:

$$\mathbf{y}_i = \sum_t \tilde{\mathbf{v}}_{it} \cdot \frac{\exp\langle \tilde{\mathbf{q}}_i, \tilde{\mathbf{k}}_{it} \rangle}{\sum_{t'} \exp\langle \tilde{\mathbf{q}}_i, \tilde{\mathbf{k}}_{it'} \rangle}. \tag{4}$$

1.4 Classification Module & classifier

The Classification Module is made up of 2 self-attention layers, each with 4 heads. The input dimension of features is 512 and the feed-forward dimension 2048. A learnable CLS token is concatenated to the input features, the output of which is then fed into a downstream classifier for final classification. The downstream classifier is an MLP with two linear layers and a tanh activation between them.

2 Hand Contact State Classification

We use the training and validation split in SomethingElse [3] for hand contact state classification. To generate 'ground truth' contact state labels, we use a pre-trained object-hand state detector from [4]. The detector predicts 5 hand contact states, namely 'no contact', 'self contact', 'other person contact', 'portable object contact' and 'stationary object contact' (e.g., furniture). It labels 85% of the frames in SomethingElse as 'portable object contact' and the rest as other types of contact. Instead of doing classification on a very unbalanced contact state, we design a 3-way classification task by categorizing the videos into the following classes:

1. No hand contact: there is no hand in the video, or there are hands in the frames but they are not in contact with any object.
2. One hand contact: There are one or two hands in the video, only one hand is in contact with objects.
3. Two hands contact: There are two hands in the video, they in contact with the same or different objects.

To do this video-level categorization, frame with the largest number of hands detected are used from each video, we check whether these hands are labelled as 'no contact' to decide which class the video falls in. The class distribution is shown in Table 1. Examples of the 3 classes are visualized in Figure 2.

Class	no contact	one hand contact	two hands contact
%videos	9%	31%	59%

Table 1. Distribution of classes in hand contract state classification in SomethingElse. We use the labels provided by a pre-trained hand state detector [4], and categorize the videos into 3 classes: 'no hand contact', 'one hand contact' and 'two hands contact'.

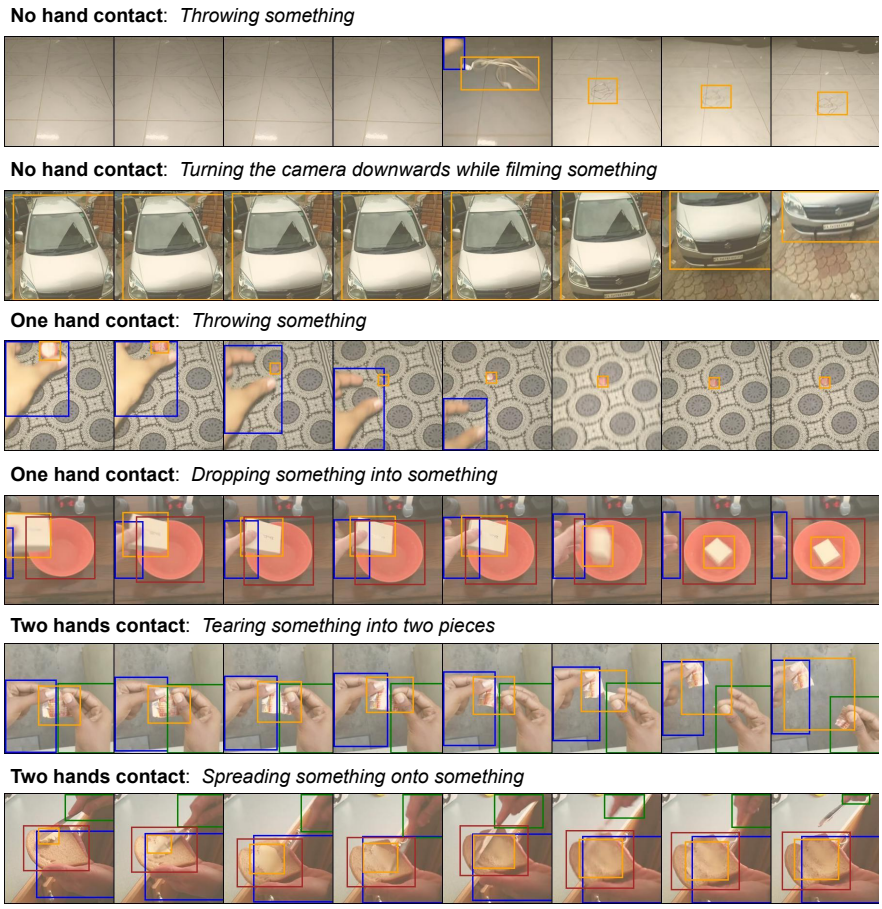

Fig. 2. Visualization of samples from the three classes in hand contact state classification. Above each video sample, we show its hand contact state class together with its action class. Ground-truth bounding boxes of hands and objects are plotted in the frames, with blue and green boxes on hands. yellow and red boxes on objects.

3 Human-Object Predicate Prediction

Given the bounding box and category of an object, the model is tasked to predict the predicate between human and this object. Action Genome [5] has annotations in 37 categories (36 objects + 1 human). There are 25 human-object relationships (aka. predicate), including 3 attention relationships, 6 spatial relationships and 16 contact relationships. There can be more than one relationships between a person and an object, thus the performance is measured in terms of recall.

When we linear probe our baselines Motionformer and Motionformer+STLT, we use the CLS tokens from the backbones, and concatenate them with a one-hot object-id, indicating which object we want to predict the predicate. When linear probing our model with an Object Learner, we use the same approach except that we are also concatenate the CLS tokens and object-ids with the addtiontal object-centric representations of the given object. We train a 25-way linear classifier on the the concatenated vector to predict the predicate classes, using a binary cross-entropy loss.

4 Ablations

4.1 Number of context queries

We ablate the number of context queries in our Object Learner. In Table 2 we show the classification performance with $\{0, 3, 6, 9\}$ context queries on SomethingElse. The top-1 accuracy increases by only 0.2% as the number of context queries goes from 0 to 9. The small impact might be due to the fact that action recognition in the dataset we use only depends on two or three key objects.

4.2 Ablation on choice of input layer from the visual backbone

We ablate the performance of models with the Object Learner reading from different layers in the visual backbone. We tried layers 6,8,12 from a Motionformer with 12 layers in total. Table 3 shows the results. While the accuracy of direct class predictions from our Object Learner does not differ too much ($\pm$0.2%), the input visual layer has a big influence on the combined results from Object-Learner and CLS token, where we averge the probability prediction from Object Learner and CLS token. The improvement on the averaged Top1 is 1.6% when using layer 6, and -0.1% when using layer 12. The monotonic drop with increase in depth suggests that earlier layer fusion is necessary for complementary results to our Object Learner.

4.3 Ablation on the depth and width of Object Learner.

We evalute the performance of our model using an Object Learner with a varying number of layers and heads. Table 4 shows the results ranging from 4 layers to 8 layers, and from 4 heads to 8 heads. Doubling the size of model only leads to 0.2% increase in top1 accuracy and 0.3% increse in top5 accuracy. It shows that learning good object-centric representations from small number of objects (within 5) does not require a very deep and wide Object Learner.

5 Other Auxiliary Losses

We experimented with other types of auxiliary losses on the object summary output from the Object Learner, always with the intention of improving the

#context queries	0	3	6	9
Top 1	73.5	73.6	73.7	**73.7**
Top 5	93.5	93.5	93.5	**93.6**

Table 2. Ablation on number of context queries in Object Learner. We evaluate the compositional action recognition performance on SomethingElse by using different number of context queries.

Input Visual Layer	OL Top1	Backbone Top1	Avg Top1
6	71.0	72.0	**73.6**
8	**71.3**	**72.3**	73.1
12	70.9	72.1	72.0

Table 3. Ablation on the Visual Input in Object Learner. We evaluate the performance of our model on compositional action recognition (unseen objects) with the Object Learner extracting features from layer 6,8,12 in the visual backbone of depth 12. Results show reading from the sixth layer yield best performance.

#Layers	# Heads	GFLOP	Top1	Top5
4	4	382	73.6	93.5
8	4	384.5	73.7	93.8
8	8	384.5	**73.8**	**93.8**

Table 4. Ablation on the depth and width of Object Learner on SomethingElse. We evalute the performance of our model using Object Learners with 4 layers and 4 heads, 8 layers and 4 heads, 8 layers and 8 heads.

modality fusion by encouraging the object queries to attend to both the modality streams. However, the results (Table 5) show that they do not help achieve a better performance on downstream tasks, hence these other auxiliary losses are not included in the main paper. We list them below to illustrate approaches that don't benefit action or hand state classification.

5.1 Instance-level contrastive loss on visual transformation vectors

To induce greater object-awareness, we train the object summary vectors to be able to pick out 'correct' visual dynamics from incorrect/synthetically generated ones. To this end, we introduce a contrastive loss with estimated 'transformation vectors', which embeds the visual affinities of objects along the temporal dimension. The 'transformer vectors' computed from frames in a correct temporal order serve as positive samples, and the ones computed from temporally shuffled frames serve as negative samples in the loss. The summary vectors are then tasked with associating each object to its 'correct' sample from the bag of positives and negatives.

More specifically, given the visual feature maps of a clip and the object bounding boxes in it, we RoI-Pool the object features w_j from each frame, where j is the index of object. Based on these per-frame object features, we compute the 'affinity vector' between frames by:

$$\tilde{\mathbf{aff}}_j^i = \mathbf{w}_j^i \cdot \mathbf{w}^{\top}{}_j^{i+1}, \tag{5}$$

$$\tilde{\mathbf{aff}}_j^{shuffle,i} = \mathbf{w}_j^i \cdot \mathbf{w}^{\top}{}_j^k, \quad k \neq i+1 \tag{6}$$

We embed the affinity vectors of an object along the temporal dimension into a transformation vector z_j. The encoding is done by using a small Transformer $g(.)$ with 2 layers and 4 heads.

$$\mathbf{z_j} = g(\tilde{\mathbf{aff}}_j), \tag{7}$$

$$\mathbf{z}_j^{shuffle} = g(\tilde{\mathbf{aff}}_j^{shuffle}), \tag{8}$$

$$\tilde{\mathbf{aff}}_j = (\tilde{\mathbf{aff}}_j^1, \tilde{\mathbf{aff}}_j^2, \ldots, \tilde{\mathbf{aff}}_j^{T'-1}) \tag{9}$$

$$\tilde{\mathbf{aff}}_j^{shuffle} = Shuffle(\tilde{\mathbf{aff}}_j^1, \tilde{\mathbf{aff}}_j^2, \ldots, \tilde{\mathbf{aff}}_j^{T'-1}) \tag{10}$$

z_j and $z_j^{shuffle}$ are used to compute the contrastive loss on object summary vectors s_j as in:

$$\mathcal{L}_{aff} = -\sum_j \left[\log \frac{\exp(s_j^{\top} \cdot z_j)}{\sum_k \exp(s_j^{\top} \cdot z_k) + \sum_k \exp(s_j^{\top} \cdot z_k^{shuffle})} \right] \tag{11}$$

5.2 Class-level contrastive loss on RoI-Pooled object vectors

Based on the hypothesis that objects under the same action may have similar transformation of states, we design a contrastive loss to push these object summaries closer in the feature space. For each object j and the action class label l it is associated with. We apply a supervised contrastive loss on each object summary vector s_j, where other vectors with the same class label l serve as its positive samples, with different class labels are used as its negative samples.

$$\mathcal{L}_{obj} = -\sum_j \left[\log \frac{\sum_k \exp(s_{j,l}^{\top} \cdot s_{k,l})}{\sum_k \exp(s_{j,l}^{\top} \cdot s_{k,l}) + \sum_{m,l' \neq l} \exp(s_{j,l}^{\top} \cdot s_{m,l'})} \right] \tag{12}$$

Loss	OL only	Backbone only	Final
L_{traj}	71.0	72.0	73.6
$L_{traj} + L_{trans}$	71.0	72.1	73.5
$L_{traj} + L_{trans} + L_{obj}$	70.0	72.2	73.2

Table 5. Ablation on different types of auxiliary losses on SomethingElse. Adding other auxiliary losses does not improve the action classification results. We choose to use a single contrastive loss on trajectories (Eq.1 in main paper) for simplicity.

6 Implementation Details

6.1 Model architecture.

We use Motionformer [1] as the visual encoder, operating on 16 frames of size 224×224 pixels uniformly sampled from a video; the 3D patch size for tokenization is 2×16×16. We use STLT [2] as the trajectory encoder which takes normalized bounding boxes from 16 frames as input. Our Object Learner is a Cross-Transformer with 6 layers and 8 heads. We adopt the trajectory attention introduced in [1] instead of the conventional joint spatio-temporal attention in the layers. The Classification Module has 4 self-attention layers with 6 heads. We set the number of context queries as 2 in all the datasets, and number of object queries as 6 in SomethingElse, SomethingSomething and EpicKitchens, 37 in ActionGenome.

6.2 Data preprocessing

During training, we sample clips of size $16 \times 224 \times 224$ uniformly from videos so that the temporal span of the clips cover the whole video. Input images are normalized with mean and standard deviation 0.5, rescaling in the range $[-1, 1]$. For data augmentation, we apply random scale jittering from scale 180 to 256 such that the objects are not cropped out of the frames, random spatial cropping at size 224×224, and random horizontal flips only to flipping-invariant classes (determined by class descriptions). We also use RandAugment [6] with maximum magnitude 20 for color jittering. For inference, we use 3-crop evaluation following previous works [1,7].

6.3 Training

We train the model with an AdamW [8] optimizer for 35 epochs with weight decay 1×10^{-3}. The base learning rate is 3.75×10^{-5}, decayed by 0.1 and 0.01 at epoch 20 and 30. We use label smoothing [9] with alpha 0.2 and mixed precision training. The rate of DropConnect [10] in all attention layers is set to 0.2. Due to limited compute resources (making joint end-to-end training infeasible), we first train the visual and trajectory backbone separately on corresponding training set, then freeze the visual backbone and fine-tune the trajectory backbone, Object Learner and Classification Module on 2 RTX 6000 GPUs with batch size 72.

7 More Visualizations

In the main paper we have shown some visualizations of object-aware attention in the Object Learner (Fig.4 in main paper), from models trained with and without auxiliary loss. The visualizations are done by plotting the attention scores from the last cross-attention layer. Here we add some more examples in Figure 3.

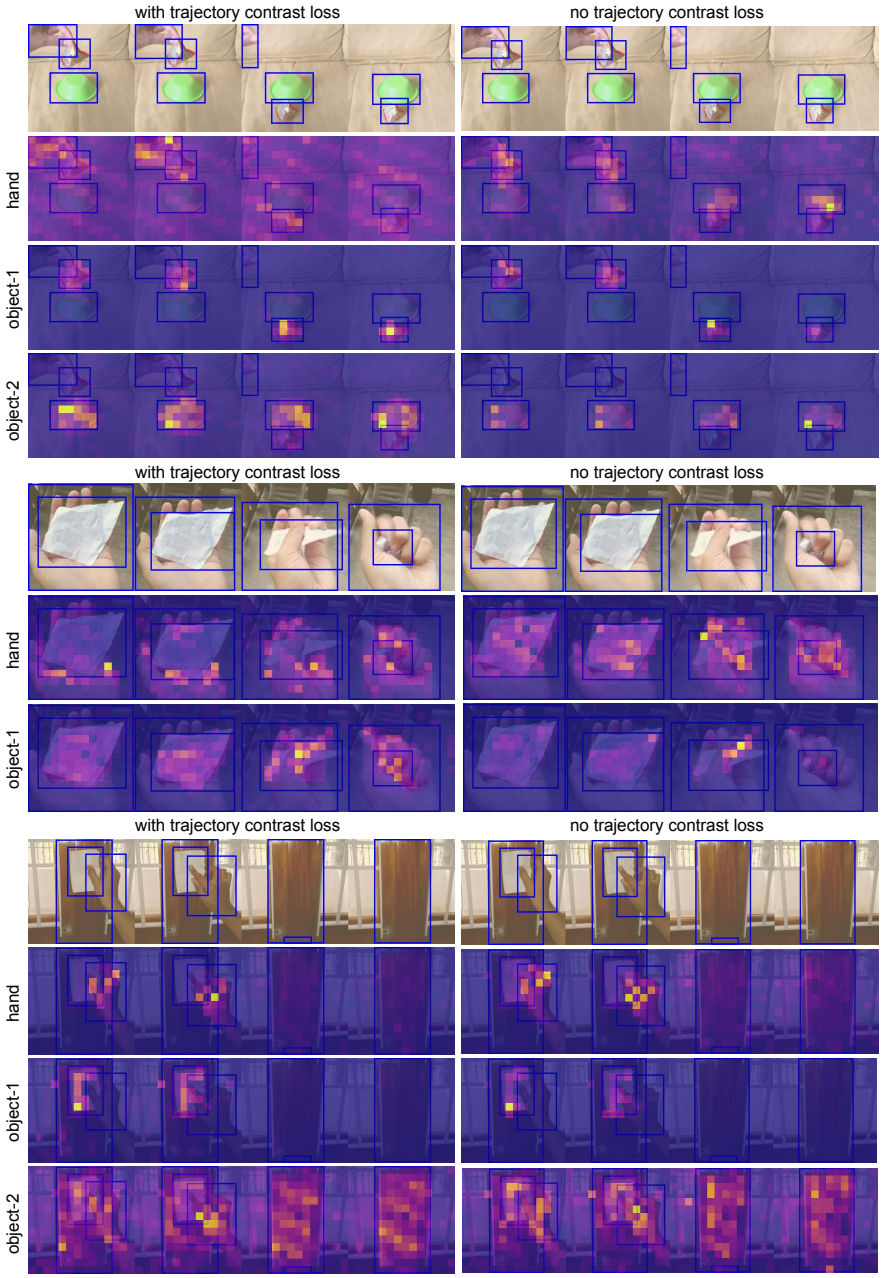

Fig. 3. Visualization of object-aware attention in Object Learner, from models trained with (Left) and without (Right) auxiliary loss. Attention of object queries on visual feature map is visualized above. Although in both cases the attention is object-centric, the one trained without auxiliary loss does not always attend to the hands (middle figure), and has either weak or peaky attention on some parts of the objects (object-1, object-2 in the upper figure, object1-in the lower figure). While the one trained with the auxiliary loss always pays attention to the hand and even has strong attention on the full objects. Brighter colors indicates higher attention scores.

References

1. Patrick, M., Campbell, D., Asano, Y., Misra, I., Metze, F., Feichtenhofer, C., Vedaldi, A., Henriques, J.F.: Keeping your eye on the ball: Trajectory attention in video transformers. NeurIPS (2021) 2, 3, 9
2. Radevski, G., Moens, M.F., Tuytelaars, T.: Revisiting spatio-temporal layouts for compositional action recognition. In: Proc. BMVC. (2021) 2, 3, 9
3. Materzynska, J., Xiao, T., Herzig, R., Xu, H., Wang, X., Darrell, T.: Something-else: Compositional action recognition with spatial-temporal interaction networks. In: Proc. CVPR. (2020) 4
4. Shan, D., Geng, J., Shu, M., Fouhey, D.F.: Understanding human hands in contact at internet scale. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2020) 9869–9878 4
5. Ji, J., Krishna, R., Fei-Fei, L., Niebles, J.C.: Action genome: Actions as compositions of spatio-temporal scene graphs. In: Proc. CVPR. (2020) 5
6. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical data augmentation with no separate search. arXiv preprint arXiv:1909.13719 **2** (2019) 7 9
7. Herzig, R., Ben-Avraham, E., Mangalam, K., Bar, A., Chechik, G., Rohrbach, A., Darrell, T., Globerson, A.: Object-region video transformers. arXiv preprint arXiv:2110.06915 (2021) 9
8. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proc. ICLR. (2019) 9
9. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2818–2826 9
10. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision, Springer (2016) 646–661 9