Supplementary Material

In this supplementary file, we provide additional information about our model, implementation details, experimental results, and qualitative examples. Specifically, Section A provides additional implementation details, Section B provides additional model details, Section C provides additional ablations of our approach, Section D provides more experiment results, and we show qualitative visualizations to demonstrate our approach in Section E.

A. Additional Implementation Details

We add ORViT to multiple existing transformer models Mformer (MF) [65], MViT [30], and TimesFormer [7]. These are all implemented based on the SlowFast [21] library (available at `https://github.com/facebookresearch/SlowFast`), and we implement ORViT based on this repository. Next, we elaborate on how we extract the object regions, and for each dataset, we add additional implementation details.

A.1. Detector and Tracker

Detector. In all action recognition datasets we used Faster R-CNN detector [31,67] with ResNet-50 backbone [32] and Feature Pyramid Network (FPN) [59] that is pre-trained on the MS COCO [60] dataset. We used the Detectron2 [86] implementation. In SSv2, the detector is finetuned using the bounding boxes annotation. During finetuning the class categories are *hand* and *object*. For AVA, we used the provided detection boxes for the spatio-temporal action detection task that were first obtained by Faster-RCNN pre-trained over MS COCO and then fine-tuned on AVA, as in [21, 84]. We set the number of objects in our model to 4 in SSv2 and EK100, 6 in AVA, and 10 in Diving48. If fewer objects are presented, we set the object coordinates with a zero vector. These numbers were chosen by taking the max number of objects (after removing the outliers) per video (as induced by the tracker output) across all videos in the training set.

Tracker. Once we have the detector results, we apply multi-object tracking to find correspondence between the objects in different frames. We use SORT [8]: a simple tracker implemented based on Kalman Filter [43] and the Hungarian matching algorithm (KM) [50]. At each step, the Kalman Filter predicts plausible instances in the current frame based on previous tracks. Next, the predictions are matched with single-frame detections by the Hungarian matching algorithm. It is important to note that the tracker does not require any training and does not use any additional data. If an object does not appear in one of the frames, we set the coordinates in these frames to zeros. We provide some stats about the accuracy of the tracking step. Based on two consecutive frames of SSv2, SORT yields an 90.1% exact match between boxes.

Choosing the O on each dataset. We mentioned that O is set to the maximum number of objects per video (generated by the tracker) across all videos in the training set. In addition, we also tested an approach that does not require setting O ahead of time: for each batch take the maximum number of objects in any clip in the batch, and pad all clips to this number. This results in a very similar performance (-0.1%). This padding approach reduces the pre-processing step of finding the numbers for any dataset by choosing a fixed and large enough number. We also measured the inference runtime (milliseconds per clip) with and without SORT, and it increased by x1.18 (from 60.9ms to 71.6ms). Additionally, the object tokens add x1.03 FLOPS and x1.05 run-time.

A.2. Something-Something v2

Dataset. The SomethingElse [26] contains 174 action categories of common human-object interactions. We follow the official splits from [26].

Optimization details. For the standard SSv2 [63] dataset, we trained 16 frames with sample rate 4 and batch-size 48 on 8 RTX 3090 GPUs. We train our network for 35 epochs with Adam optimizer [47] with a momentum of 0.9 and Gamma 0.1. Following [65], we use $lr = 5e-5$ with $\times 10$ decay steps at epochs $0, 20, 30$. Additionally, we used Automatic Mixed Precision, which is implemented by PyTorch. We initialize from a Kinetics-400 pre-trained model [46]. For the *ORViT MF-L* model, we fine-tuned from the SSv2 pre-trained model provided by [65] and train with 32 frames. The optimization policy is similar to the above, except we used a different learning rate: $1e-5$ for the pre-trained parameters, and $1e-4$ for the ORViT parameters.

For the compositional action recognition task, we trained on the SomethingElse splits [63]. We train with a batch size of 32 and a learning rate of $3e-5$.

Regularization details. We use weight decay of 0.05, a dropout [36] of 0.5 before the final classification, dropout of 0.3 after the ORViT block, and DropConnect [37] with rate 0.2.

Training details. We use a standard crop size of 224, and we jitter the scales from 256 to 320. Additionally, we use RandAugment [14] with maximum magnitude 20 for each frame separately.

Inference details. We take 3 spatial crops per single clip to form predictions over a single video in testing as done in [65].

A.3. EpicKitchens100

Dataset. EK100 [16] contains 700 egocentric videos of daily kitchen activities. This dataset includes 300 noun and 97 verb classes, and we report verb, noun, and action top-1 accuracy, while the highest-scoring of the verb and noun pairs constitutes the action label.

Optimization details. We trained over videos of 16 frames with sample rate 4 and batch-size 16 on 8 Quadro RTX 8000 GPUs. We train our network for 35 epochs with Adam optimizer [47] with a momentum of 0.9 and Gamma 0.1. Following [65], we use $lr = 5e-5$ with $\times 10$ decay steps at epochs $0, 30, 40$. Additionally, we used Automatic Mixed Precision, which is implemented by PyTorch. We initialize from a Kinetics-400 pre-trained model [46].
Training details. We use crop size of 336 for the *ORViT MF-HR*. We jitter the scales from 384 to 480. Additionally, we use RandAugment [14] with maximum magnitude 20.
Inference details. We take 3 spatial crops with 10 different clips sampled randomly to aggregate predictions over a single video in testing.

A.4. Diving48

Dataset. Diving48 [54] contains 16K training and 3K testing videos spanning 48 fine-grained diving categories of diving activities. For all of these datasets, we use standard classification accuracy as our main performance metric.
Optimization details. We trained over videos of 32 frames with sample rate 8 and batch-size 8 on 8 Quadro RTX 8000 GPUs. We train our network for 35 epochs with Adam optimizer [47] with a momentum of 0.9 and Gamma 0.1. We use $lr = 3.75e{-}5$ with $\times 10$ decay steps at epochs $0, 20, 30$. Additionally, we used Automatic Mixed Precision, which is implemented by PyTorch. We initialize from a Kinetics-400 pre-trained model [46].
Training details. We use a standard crop size of 224 for the standard model and jitter the scales from 256 to 320. Additionally, we use RandomFlip augmentation. Finally, we sampled the T frames from the start and end diving annotation time, followed by [97].
Inference details. We take 3 spatial crops per single clip to form predictions over a single video in testing same as in [7].
Object-Dynamics Module. As we show in Table 5d, we compared different self-attention mechanisms, and the standard self-attention usually performed better. However, we observe a slight improvement when we perform a trajectory self-attention [65] instead of the standard self-attention.

A.5. AVA

Architecture. SlowFast [21] and MViT [30] are using a detection architecture with a RoI Align head on top of the spatio-temporal features. We followed their implementation to allow a direct comparison. Next we elaborate on the RoI Align head proposed in SlowFast [21]. First, we extract the feature maps from our ORViT MViT model by using the RoIAlign layer. Next, we take the 2D proposal at a frame into a 3D RoI by replicating it along the temporal axis, followed by a temporal global average pooling. Then, we max-pooled the RoI features and fed them to an MLP classifier for prediction.
Optimization details. To allow a direct comparison, we used the same configuration as in MViT [30]. We trained 16 frames with sample rate 4, depth of 16 layers and batch-size 32 on 8 RTX 3090 GPUs. We train our network for 30 epochs with an SGD optimizer. We use $lr = 0.03$ with a weight decay of $1e-8$ and a half-period cosine schedule of learning rate decaying. We use mixed precision and fine-tune from an MViT-B, 16×4 pre-trained model.
Training details. We use a standard crop size of 224 and we jitter the scales from 256 to 320. We use the same ground-truth boxes and proposals that overlap with ground-truth boxes by $IoU > 0.9$ as in [21].
Inference details. We perform inference on a single clip with 16 frames. For each sample, the evaluation frame is centered in frame 8. We use a crop size of 224 in test time. We take 1 spatial crop with 10 different clips sampled randomly to aggregate predictions over a single video in testing.

A.6. Few Shot Compositional Action Recognition

We also evaluate on the few-shot compositional action recognition task in [63]. For this setting, we have 88 *base* action categories and 86 *novel* action categories. We train on the base categories (113K/12K for training/validation) and finetune on few-shot samples from the novel categories (for 5-shot, 430/50K for training/validation; for 10-shot, 860/44K for training/validation).

A.7. SomethingElse

Dataset. The SomethingElse [63] contains 174 action categories with 54,919 training and 57,876 validation samples. The proposed compositional [63] split in this dataset provides disjoint combinations of a verb (action) and noun (object) in the training and testing sets. This split defines two disjoint groups of nouns $\{\mathcal{A}, \mathcal{B}\}$ and verbs $\{1, 2\}$. Given the splits of groups, they combine the training set as $1\mathcal{A} + 2\mathcal{B}$, while the validation set is constructed by flipping the combination into $1\mathcal{B} + 2\mathcal{A}$. In this way, different combinations of verbs and nouns are divided into training or testing splits.
Optimization details. We trained 16 frames with sample rate 4 and batch-size 32 on 8 RTX 3090 GPUs. We train our network for 35 epochs with Adam optimizer [47] with a momentum of 0.9 and Gamma 0.1. We use $lr = 3e-5$ with $\times 10$ decay steps at epochs $0, 20, 30$. Additionally, we used Automatic Mixed Precision, which is implemented by PyTorch. We initialize from a Kinetics-400 pre-trained model [46].
Regularization details. We use weight decay of 0.05, a dropout [36] of 0.5 before the final classification, dropout of 0.3 after the ORViT block, and DropConnect [37] with rate 0.2.

(a) **Streams**

Model	Top-1	Top-5	GFLOPs (10^9)	Param (10^6)
Baseline	60.2	85.5	$\times$1 (369.5)	$\times$1 (109)
+ Object-Region Attention	67.4	89.8	$\times$1.03 (382)	$\times$1.02 (111)
+ Object-Dynamics Module	**69.7**	**91.0**	$\times$1.1 (405)	$\times$1.36 (148)

(b) **Blocks Ablation**

Layers	Top-1	Top-5	GFLOPs (10^9)	Param (10^6)
Baseline	60.2	85.8	$\times$1 (369.5)	$\times$1 (109)
2	68.9	90.4	$\times$1.03 (381)	$\times$1.12 (122)
7	67.8	89.5	$\times$1.03 (381)	$\times$1.12 (122)
11	66.8	89.3	$\times$1.03 (381)	$\times$1.12 (122)
2, 7	69.3	90.6	$\times$1.06 (393)	$\times$1.24 (135)
2, 7, 11	**69.7**	**91.0**	$\times$1.1 (405)	$\times$1.36 (148)

(c) **Object-Region Attention**

Model	Top-1	Top-5
Baseline	60.2	85.8
Ours /w Joint attention	68.9	90.4
Ours /w Divided attention	69.3	90.6
Ours /w Trajectory attention	**69.7**	**91.0**

(d) **Object-Dynamic Module**

Model	Top-1	Top-5
GCN	67.7	89.8
Trajectory attention	69.4	90.5
Self-attention	**69.7**	**91.0**

(e) **Components**

Layers	Top-1	Param
Baseline	80.0	$\times$1 (121)
ORViT [:12]	85.4	$\times$1.01 (122)
ORViT [:2]	86.8	$\times$1.01 (122)
ORViT [:2] + ODM	87.5	$\times$1.11 (134)
ORViT [:2, 7, 11] + ODM	**88.0**	$\times$1.32 (160)

Table 5. **Ablations.** Evaluation of different model ablations and baselines on the "SomethingElse" split (Tables (a-d) see text). We report pretrain, param (10^6), GFLOPS (10^9), and top-1 and top-5 video action recognition accuracy. Table (e) reports ablations on the Diving dataset.

Training details. We use a standard crop size of 224, and we jitter the scales from 256 to 320.

Inference details. We take 3 spatial crops per single clip to form predictions over a single video in testing.

B. Additional Model details

B.1. Object-Region Attention

As explained in section 3.2, there are two inputs to the ORViT block. The first is the output of the preceding transformer block, represented as a set of spatio-temporal tokens $X \in \mathbb{R}^{THW \times d}$. The second input is a set of bounding boxes for objects across time, denoted by $B \in \mathbb{R}^{TO \times 4}$.

Object-Region Attention. Given the patch token features X and the boxes B, we use RoIAlign [31] layer, which uses the patch tokens X and box coordinates B to obtain object region crops. This is followed by max-pooling and an MLP. To these features we add a learnable object-time position encoding $\mathcal{P} \in \mathbb{R}^{TO \times d}$ to encode the positional object information. We also use a coordinate embedding by applying an MLP on the boxes coordinates, resulting in $\mathcal{L} \in \mathbb{R}^d$:

$$\mathcal{L} := \text{MLP}(B) \tag{4}$$

where $B \in \mathbb{R}^{T \times O \times d}$ is the boxes coordinates. This leads to an improved object features:

$$\mathcal{O} := \text{MLP}(\text{MaxPool}(\text{RoIAlign}(X, B))) + \mathcal{L} + \mathcal{P} \tag{5}$$

where the token features are $X \in \mathbb{R}^{THW \times d}$. We pass these features into the self-attention layers as explained in the "Object-Region attention" subsection in the main paper.

C. Additional Ablations

We perform an ablation study of each of the components in Table 5 to show the effectiveness of the different components of our model. All ablations are on the SomethingElse [63] dataset and we use Mformer (MF) as the baseline architecture for ORViT unless stated otherwise. We also note that we refer to the "Object-Dynamics Module" as ODM stream.

Contribution of appearance and motion streams. In Table 5a, we show the "Object-Region Attention" is an important factor for the improvement, responsible for a 7.2% gain improvement, with less than 2% additional parameters over MF (only 2M parameters addition compared to the baseline). This highlights our contribution that object interactions are indeed crucial for video transformers. Additionally, adding trajectory information with coordinates in the "Object-Dynamics Module" (ODM) improved by another 2.3% but with a cost of 36% additional parameters. We show later (see in Section D.1) that we can reduce the ODM size with smaller dimensions.

ORViT blocks. In Table 5b, we show which layers are most important for adding the ORViT block. The experiments show that adding this information at the network's beginning, middle, and end is the most effective (layer 2, 7, 11). This experiment demonstrates that it is important to fuse object-centric representations starting from early layers and propagate them into the transformer-layers, thus affecting the spatio-temporal representations throughout the network.

Different self-attention in "Object-Region Attention". In Table 5c, we compared different self-attention mechanisms (as defined in [65]): joint space-time, divided space-time, and trajectory attention to the *MF* baseline, which uses trajectory attention in all layers. We observed that trajectory

Dataset	Model	ODM Dimension	Top-1	Top-5	GFLOPs (10^9)	Param (10^6)
SomethingElse	Baseline	-	60.2	85.8	$\times$1 (369.5)	$\times$1 (109)
	ORViT	128	68.7	90.3	$\times$1.03 (382)	$\times$1.03 (112)
		256	68.9	90.5	$\times$1.04 (383)	$\times$1.05 (114)
		768	69.7	91.0	$\times$1.1 (405)	$\times$1.36 (148)
SSv2	Baseline	-	66.5	90.1	$\times$1 (369.5)	$\times$1 (109)
	ORViT	128	67.2	90.4	$\times$1.03 (382)	$\times$1.03 (112)
		256	67.3	90.5	$\times$1.04 (383)	$\times$1.05 (114)
		768	67.9	90.5	$\times$1.1 (405)	$\times$1.36 (148)

Table 6. **A light-weight version of ORViT**.

attention is slightly better. However, it can be seen that our object region approach is not sensitive to these choices, indicating that the generic approach is the main reason for the observed improvements.

Replacing ORViT with Trajectory Attention. We observe that joint and divided self-attention layers [2, 7] have similar results to the trajectory attention [65], as seen in Table 5c. However, we would like to demonstrate that trajectory attention is not the main reason for the improvement when using ORViT with TimeSformer [7] or MViT [30]. Thus, we replace our ORViT with a standard trajectory attention on the Diving48 and AVA datasets. The top1 accuracy on Diving48 are improved by 4.5% (from 80.0 to 84.5) with trajectory attention, while using our *ORViT+TimeSformer* achieves 88.0 (3.5% improvements on top of that). The MAP on AVA are the same as the baseline with trajectory attention (25.5), while using our *ORViT+MViT-B* achieves 26.6 (1.1 improvements on top of the baseline). We note that our MF is the baseline on EK100, SSv2, and SomethingElse, and therefore the trajectory attention is already part of the model, and hence this demonstration is not needed.

Processing trajectory information. In Table 5d, we compared our self-attention (see "Object-Dynamics Module" in Section 3.2) with other standard baseline models: GCN [48] and trajectory self-attention [65]. For the GCN, we use a standard implementation with 2 hidden layers, while for the trajectory attention, we treat the O objects as the spatial dimension. We can see that self-attention is slightly better than trajectory self-attention (+0.3%) and significantly better than GCN (+2.0%).

Components on Diving48. Following our components ablations in Table 4a, we also validate our hypothesis on the Diving48 dataset. We used the TimeSformer [7] trained on videos of 32 frames as a baseline for a fair comparison. It can be seen that a single layer version of the model already results in considerable improvement (85.4%) and that it is important to apply it in the earlier layers of transformers than at the end (86.8% compared to 85.4%). Additionally, the "Object-Dynamics Module" improves performance to 87.5%. Finally, multiple applications of the layer further improve performance to 88.0%.

Box Position Encoder. Our "Box Position Encoder" transforms from a tensor of size TO to size THW. Our implementation of this transformation uses box information so that each object is mapped to the "correct" region in space. A simpler approach would have been to expand the shape of TO to THW without using boxes. We refer to the latter as a standard tensor expansion. Comparing the two methods, we find out that our approach obtains 69.7 compared to 68.4, showing that our box-based encoding performs better.

Combine the two streams in ORViT Block. As part of the ORViT block, we examined other operations to combine the two streams of information. We explored the following methods: element-wise multiplication, gating (with conv), and our simple sum. The results on SomethingElse are 68.8, 68.7, and 69.7, respectively. In this case, our simple sum is superior to the other methods.

$T \times O$ **learnable embeddings**. We observe a small difference when experimenting with $T \times O$ and separate T and O embeddings (69.6 Vs. 69.7) on the SomethingElse dataset.

D. Additional Experiments

Here we present additional experiments, including demonstrating a lightweight version of the "Object-Dynamics Module" that significantly reduces the model parameters without losing significant performance and complete results on the standard action recognition task.

D.1. Light-weight ORViT

In Table 5a, we show that the "Object-Dynamics Module" improves by 2.3% the top-1 accuracy with an additional 39M parameters (148M Vs. 109M). We would like to demonstrate that model size can be significantly decreased, incurring a small performance loss. Most the parameters added by ORViT over the baseline MF are in the ODM, and thus it is possible to use a smaller embedding dimension in ODM. Here, we present a light-weight version of the module that reduces the embedding dimensions without losing significant accuracy. See Table 6.

As mentioned in the main paper (see Section 3), we use $\widetilde{B}$ for the coordinate embeddings in the "Object-Dynamics

Module". We observe that reducing the dimension of the coordinate embeddings ($\widetilde{B}$) from 768 to 256 has little impact on the action accuracy in SSv2 (67.9% Vs. 67.3%) and SomethingElse (69.7% Vs. 68.9%), although having only 114M model parameters (an addition of 5M parameters to the MF baseline that has 109M). Indeed this indicates that our main approach is the main reason for the observed improvements and not necessarily the addition of parameters.

D.2. Standard Action Recognition Results

We next report in Table 7 the full results table for the standard action recognition task, including extra models and details, which were not included in the main paper.

Additionally, we add a light version of ORViT for each dataset. This version use embedding dimension of 256 in the "Object-Dynamics Module", as stated in Section D.1. In SSv2, the ORViT-Light model improves the MF baseline by 0.8 at the cost of additional $5M$ parameters (5% more parameters), while the ORViT model (non-light version) improves by 1.4% at the cost of additional $39M$ parameters (36% more parameters). In Diving48, the ORViT-Light model improves the TimeSformer baseline by 6.8 at the cost of additional $5M$ parameters (3% more parameters), while the ORViT model (non-light version) improves by 8% at the cost of additional $39M$ parameters (32% more parameters). In EK100, the ORViT-Light model improves the MF-HR baseline by $1.6, 1.5, 0.2$ (A, V, N) at the cost of additional $5M$ parameters (5% more parameters), while the ORViT model (non-light version) improves by $1.2, 1.4, 0.2$ at the cost of additional $39M$ parameters (36% more parameters). We note that the ORViT-Light even outperforms the non-light version (1.6 Vs. 1.2), demonstrating the object movement is less significant in this data.

Last, we separately evaluated the accuracy on Kinetics-400 with ORViT MViT-B 16x4 and noticed that it improved by +2.0% over MViT-B 16x4.

E. Qualitative Visualizations

To provide insight into the inner representation of ORViT we provide further visualization next. See Figure 5 and Figure 6. In Figure 5, we visualize the attention map of the CLS token on all spatial tokens. It can be seen that object-regions indeed affect these spatial maps. For example, "Tearing something into two pieces" (top left corner) demonstrates that *ORViT+MF* successfully separates the two pieces of the paper, while the *MF* baseline does not. Next, in Figure 6 we visualize the attention allocated to each of the object keys. It can be seen that the object keys in *ORViT* indeed affect their corresponding spatial tokens.

Table 7. **Comparison to the state-of-the-art on video action recognition.** We report pretrain, param (10^6), GFLOPS (10^9) and top-1 (%) and top-5 (%) video action recognition accuracy on SSv2. On Epic-Kitchens100 (EK100), we report top-1 (%) action (A), verb (V), and noun (N) accuracy. On Diving48 we report pretrain, number of frames, param (10^6) and top-1 (%) video action recognition accuracy. Difference between baselines (MF/MF-L for SSv2, TimeSformer for Diving48, MF-HR for EK100) and ORViT is denoted by (+X). We denote methods that do not use bounding boxes with †.

(a) **Something–Something V2**

Model	Pretrain	Top-1	Top-5	GFLOPs×views (10^9)	Param (10^6)
SlowFast, R50†	K400	61.7	87.0	65.7×3×1	34.1
SlowFast, R101†	K400	63.1	87.6	106×3×1	53.3
TSM†	K400	63.4	88.5	62.4×3×2	42.9
STM†	IN-1K	64.2	89.8	66.5×3×10	-
MSNet†	IN-1K	64.7	89.4	67×1×1	24.6
TEA†	IN-1K	65.1	-	70×3×10	-
bLVNet†	IN-1K	65.2	90.3	128.6×3×10	-
VidTr-L†	IN-21K+K400	60.2	-	351×3×10	-
TimeSformer-L†	IN-21K	62.5	-	1703×3×1	121.4
ViViT-L†	IN-21K+K400	65.4	89.8	3992×4×3	-
MViT-B, 32†	K400	67.1	90.8	170×3×1	36.6
MViT-B, 64†	K400	67.7	90.9	455×3×1	36.6
MViT-B, 32†	K600	67.8	91.3	170×3×1	36.6
MViT-B, 64†	K600	68.7	91.5	236×3×1	53.2
MF†	IN-21K+K400	66.5	90.1	369.5 × 3 × 1	109
MF-L†	IN-21K+K400	68.1	91.2	1185.1 × 3 × 1	109
MF+STRG	IN+K400	66.1	90.0	375 × 3 × 1	117
MF+STIN	IN+K400	66.5	89.8	375.5 × 3 × 1	111
MF+STRG+STIN	IN+K400	66.6	90.0	375.5 × 3 × 1	119
ORViT MF-Light (Ours)	IN-21K+K400	**67.3** (+0.8)	90.5 (+0.4)	383 × 3 × 1	114 (+5%)
ORViT MF (Ours)	IN-21K+K400	**67.9** (+1.4)	90.5 (+0.4)	405 × 3 × 1	148 (+36%)
ORViT MF-L (Ours)	IN-21K+K400	**69.5** (+1.4)	**91.5** (+0.3)	1259 × 3 × 1	148 (+36%)

(b) **Diving48**

Model	Pretrain	Frames	Top-1	Params (10^6)
I3D†	K400	8	48.3	-
TSM†	ImageNet	3	51.1	42.9
TSN†	ImageNet	3	52.5	-
GST-50†	ImageNet	8	78.9	-
ST-S3D†	K400	8	50.6	-
SlowFast, R101†	K400	16	77.6	53.3
TimeSformer†	IN-21K	16	74.9	121
TimeSformer-HR†	IN-21K	16	78.0	121
TimeSformer-L†	IN-21K	96	81.0	121
TQN†	K400	ALL	81.8	-
TimeSformer†	IN-21K	32	80.0	121
TimeSformer + STIN	IN-21K	32	81.0	123
TimeSformer + STRG	IN-21K	32	78.1	129
TimeSformer + STRG + STIN	IN-21K	32	83.5	132
ORViT TimeSformer-Light (Ours)	IN-21K	32	**86.8** (+6.8)	126 (+3%)
ORViT TimeSformer (Ours)	IN-21K	32	**88.0** (+8.0)	160 (+32%)

(c) **Epic-Kitchens100**

Method	Pretrain	A	V	N	Params (10^6)
TSN†	IN-1K	33.2	60.2	46.0	-
TRN†	IN-1K	35.3	65.9	45.4	-
TBN†	IN-1K	36.7	66.0	47.2	-
TSM†	IN-1K	38.3	67.9	49.0	-
SlowFast†	K400	38.5	65.6	50.0	-
TimeSformer†	IN-21K	32.9	55.8	50.1	121
ViViT-L	IN-21K+K400	44.0	66.4	56.8	-
MF†	IN-21K+K400	43.1	66.7	56.5	109
MF-L†	IN-21K+K400	44.1	67.1	57.6	109
MF-HR†	IN-21K+K400	44.5	67.0	58.5	109
MF-HR + STIN	IN-21K+K400	44.2	67.0	57.9	111
MF-HR + STRG	IN-21K+K400	42.5	65.8	55.4	117
MF-HR + STRG + STIN	IN-21K+K400	44.1	66.9	57.8	119
ORViT MF-HR	IN21K+K400	**45.7** (+1.2)	**68.4** (+1.4)	**58.7** (+.2)	148 (+36%)
ORViT MF-HR-Light	IN21K+K400	**46.1** (+1.6)	**68.5** (+1.5)	**58.7** (+.2)	114 (+5%)

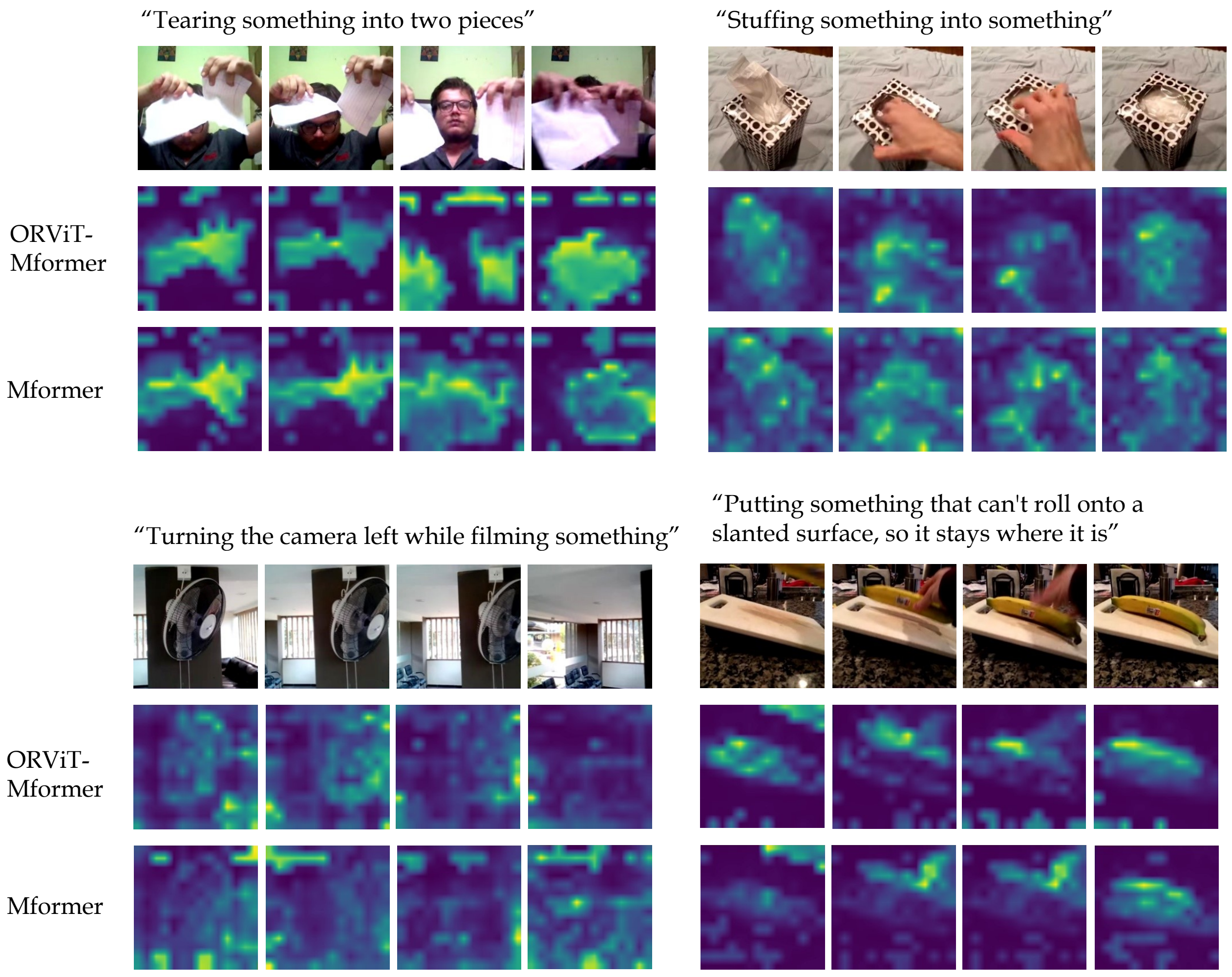
“Tearing something into two pieces”
“Stuffing something into something”
ORViT-Mformer
Mformer
“Turning the camera left while filming something”
“Putting something that can't roll onto a slanted surface, so it stays where it is”
ORViT-Mformer
Mformer

Figure 5. **Attention Maps** comparison between the *ORViT+MF* and the *MF* on videos from the SSv2 dataset. The visualization shows the attention maps corresponding to the CLS query.

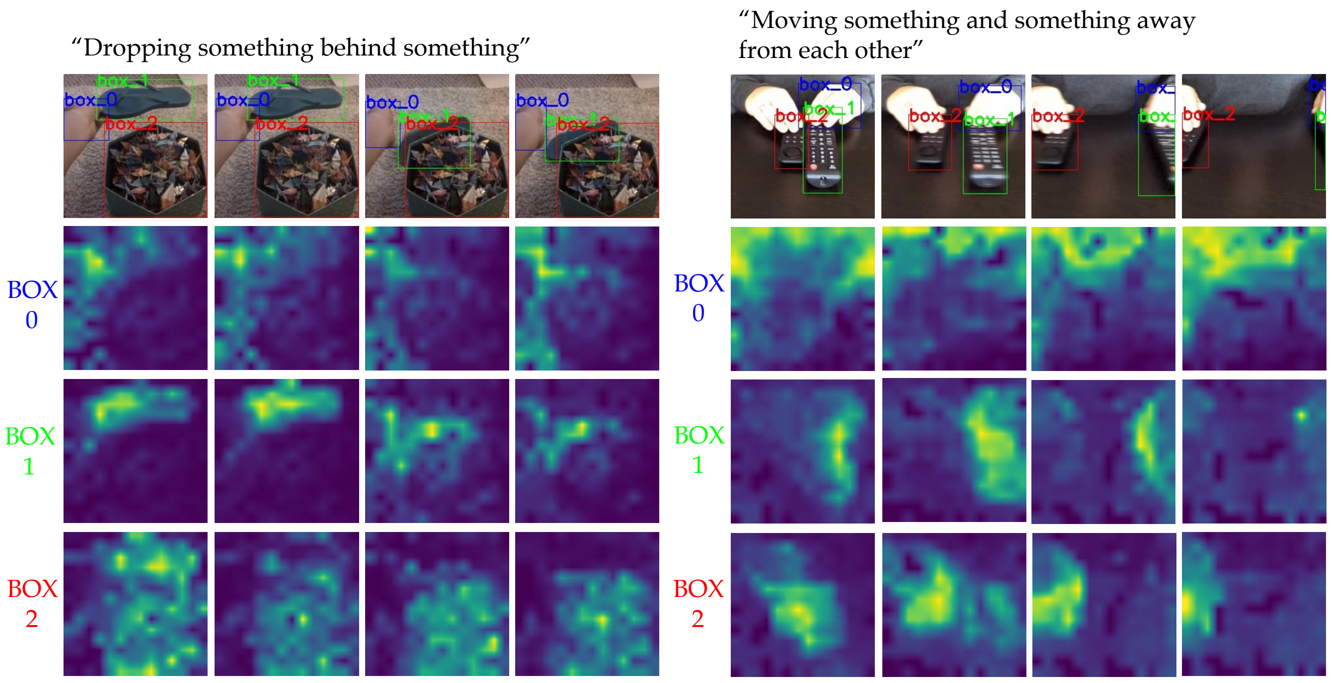

Figure 6. **Object contribution to the patch tokens.** For each object token, we plot the attention weight given by the patch tokens, normalized over the patch tokens.