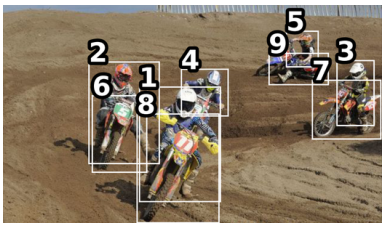
Efficient Two-Stage Detection of Human–Object Interactions with a Novel Unary–Pairwise Transformer

Frederic Z. Zhang[1,3] Dylan Campbell[2,3] Stephen Gould[1,3]
[1]The Australian National University [2]University of Oxford
[3]Australian Centre for Robotic Vision

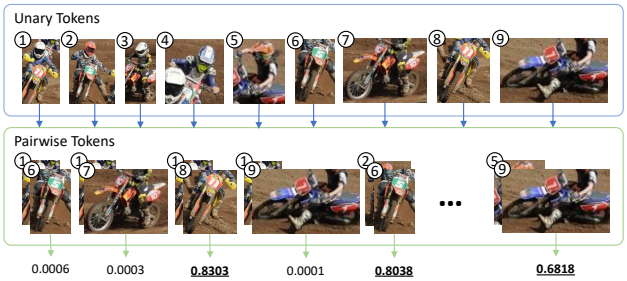
https://fredzzhang.com/unary-pairwise-transformers

Abstract

Recent developments in transformer models for visual data have led to significant improvements in recognition and detection tasks. In particular, using learnable queries in place of region proposals has given rise to a new class of one-stage detection models, spearheaded by the Detection Transformer (DETR). Variations on this one-stage approach have since dominated human–object interaction (HOI) detection. However, the success of such one-stage HOI detectors can largely be attributed to the representation power of transformers. We discovered that when equipped with the same transformer, their two-stage counterparts can be more performant and memory-efficient, while taking a fraction of the time to train. In this work, we propose the Unary–Pairwise Transformer, a two-stage detector that exploits unary and pairwise representations for HOIs. We observe that the unary and pairwise parts of our transformer network specialise, with the former preferentially increasing the scores of positive examples and the latter decreasing the scores of negative examples. We evaluate our method on the HICO-DET and V-COCO datasets, and significantly outperform state-of-the-art approaches. At inference time, our model with ResNet50 approaches real-time performance on a single GPU.



(a) Image with human and object detections.



(b) Unary and pairwise tokens with predicted scores (*riding a motorcycle*).

Figure 1. Our Unary–Pairwise Transformer encodes human and object instances individually and in pairs, allowing it to reason about the data in complementary ways. In this example, our network correctly identifies the interactive pairs for the action *riding a motorcycle*, while suppressing the visually-similar non-interactive pairs and those with different associated actions.

1. Introduction

Human–object interaction (HOI) detectors localise interactive human–object pairs in an image and classify the actions. They can be categorised as one- or two-stage, mirroring the grouping of object detectors. Exemplified by Faster R-CNN [24], two-stage object detectors typically include a region proposal network, which explicitly encodes potential regions of interest in the form of bounding boxes. These bounding boxes can then be classified and further refined via regression in a downstream network. In contrast, one-stage detectors, such as RetinaNet [18], retain the abstract feature representations of objects throughout the network, and decode them into bounding boxes and classification scores at the end of the pipeline.

In addition to the same categorisation convention, HOI detectors need to localise two bounding boxes per instance instead of one. Early works [2, 8, 16, 23] employ a pre-trained object detector to obtain a set of human and object boxes, which are paired up exhaustively and processed by a downstream network for interaction classification. This methodology coincides with that of two-stage detectors and quickly became the mainstream approach due to the accessibility of high-quality pre-trained object detectors. The

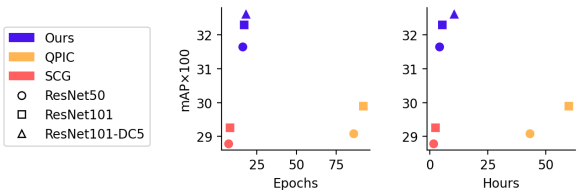


Figure 2. Mean average precision as a function of the number of epochs (left) and training time (right) to convergence. The backbone networks for all methods have been initialised with the same weights and trained on 8 GeForce GTX TITAN X GPUs.

Table 1. The performance discrepancy between existing state-of-the-art one-stage and two-stage HOI detectors is largely attributable to the choice of backbone network. We report the mean average precision ($\times 100$) on the HICO-DET [2] test set.

Method	Type	Detector Backbone	mAP
SCG [28]	two-stage	Faster R-CNN R-50-FPN	24.88
SCG [28]	two-stage	DETR R-50	28.79
SCG [28]	two-stage	DETR R-101	**29.26**
QPIC [25]	one-stage	DETR R-50	29.07
QPIC [25]	one-stage	DETR R-101	**29.90**
Ours	two-stage	DETR R-50	31.66
Ours	two-stage	DETR R-101	**32.31**

first instance of one-stage HOI detectors was introduced by Liao et al. [17]. They characterised human–object pairs as interaction points, represented as the midpoint of the human and object box centres. Recently, due to the great success in using learnable queries in transformer decoders for localisation [1], the development of one-stage HOI detectors has been greatly advanced. However, HOI detectors that adapt the DETR model rely heavily on the transformer, which is notoriously difficult to train [20], to produce discriminative features. In particular, when initialised with DETR's pre-trained weights, the decoder attends to regions of high objectness by default. The heavy-weight decoder stack then has to be adapted to attend to regions of high interactiveness. Consequently, training such one-stage detectors often consumes large amounts of memory and time as shown in Fig. 2. In contrast, two-stage HOI detectors do not repurpose the backbone network, but maintain it as an object detector. Since the first half of the pipeline already functions as intended at the beginning of training, the second half can be trained quickly for the specific task of HOI detection. Furthermore, since the object detector can be decoupled from the downstream interaction head during training, its weights can be frozen, and a lighter-weight network can be used for interaction detection, saving a substantial amount of memory and computational resources.

Despite these advantages, the performance of two-stage detectors has lagged behind their one-stage counterparts. However, most of these two-stage models used Faster R-CNN [24] rather than more recent object detectors. We found that simply replacing Faster R-CNN with the DETR model in an existing two-stage detector (SCG) [28] resulted in a significant improvement, putting it on par with a state-of-the-art one-stage detector (QPIC), as shown in Tab. 1. We attribute this performance gain to the representation power of transformers and bipartite matching loss [1]. The latter is particularly important because it resolves the misalignment between the training procedure and evaluation protocol. The evaluation protocol dictates that, amongst all detections associated with the same ground truth, the highest scoring one is the true positive while the others are false positives. Without bipartite matching, all such detections will be labelled as positives. The detector then has to resort to heuristics such as non-maximum suppression to mitigate the issue, resulting in procedural misalignment.

We propose a two-stage model that refines the output features from DETR with additional transformer layers for HOI classification. As shown in Fig. 1, we encode the instance information in two ways: a unary representation where individual human and object instances are encoded separately, and a pairwise representation where human–object pairs are encoded jointly. These representations provide orthogonal information, and we observe different behaviours in their associated layers. The unary encoder layer preferentially increases the predicted interaction scores for positive examples, while the pairwise encoder layer suppresses the negative examples. As a result, this complementary behaviour widens the gap between scores of positive and negative examples, particularly benefiting ranking metrics such as mean average precision (mAP).

Our primary contribution is a novel and efficient two-stage HOI detector with unary and pairwise encodings. Our secondary contribution is demonstrating how pairwise box positional encodings—critical for HOI detection—can be incorporated into a transformer architecture, enabling it to jointly reason about unary appearance and pairwise spatial information. We further provide a detailed analysis on the behaviour of the two encoder layers, showing that they have complementary properties. Our proposed model not only outperforms state-of-the-art methods, but also consumes much less time and memory to train. The latter allows us to employ more memory-intensive backbone networks, further improving the performance.

2. Related work

Transformer networks [27], initially developed for machine translation, have recently become ubiquitous in computer vision due to their representation power, flexibility, and global receptive field via the attention mechanism. The image transformer ViT [4] represented an image as a set of spatial patches, each of which was encoded as a token through simple linear transformations. This approach

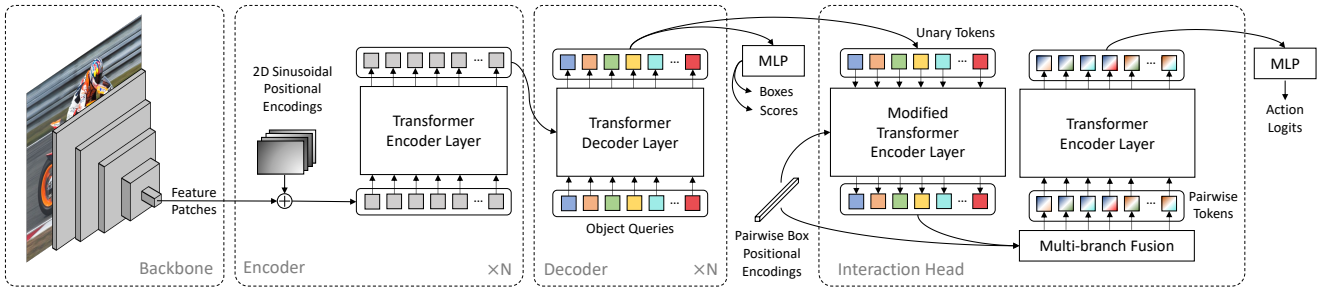


Figure 3. Flowchart for our unary–pairwise transformer. An input image is processed by a backbone CNN to produce image features, which are partitioned into patches of equal size and augmented with sinusoidal positional encodings. These tokens are fed into the DETR [1] transformer encoder–decoder stack, generating new features for a fixed number of learnable object queries. These are decoded by an MLP as object classification scores and bounding boxes, and are also passed to the interaction head as unary tokens. The interaction head also receives pairwise positional encodings computed from the predicted bounding box coordinates. A modified transformer encoder layer then refines the unary tokens using the pairwise positional encodings. The output tokens are paired up and fused with the same positional encodings to produce pairwise tokens, which are processed by a standard transformer encoder layer before an MLP decodes the final features as action classification scores.

for tokenising images rapidly gained traction and inspired many subsequent works [21]. Another key innovation of transformers is the use of learnable queries in the decoder, which are initialised randomly and updated through alternating self-attention and cross-attention with encoder tokens. Carion et al. [1] use these as object queries in place of conventional region proposals for their object detector. Together with a bipartite matching loss, this design gave rise to a new class of one-stage detection models that formulate the detection task as a set prediction problem. It has since inspired numerous works in HOI detection [3, 13, 25, 29].

To adapt the DETR model to HOI detection, Tamura et al. [25] and Zou et al. [29] add additional heads to the transformer in order to localise both the human and object, as well as predict the action. As for bipartite matching, additional cost terms are added for action prediction. On the other hand, Kim et al. [13] and Chen et al. [3] propose an interaction decoder to be used alongside the DETR instance decoder. It is specifically responsible for predicting the action while also matching the interactive human–object pairs. These aforementioned one-stage detectors have achieved tremendous success in pushing the state-of-the-art performance. However, they all require significant resources to train the models. In contrast, this work focuses on exploiting novel ideas to produce equally discriminative features while preserving the memory efficiency and low training time of two-stage detectors.

Two-stage HOI detectors have also undergone significant development recently. Li et al. [15] studied the integration and decomposition of HOIs in an analogy to the superposition of waves in harmonic analysis. Hou et al. explored few-shot learning by fabricating object representations in feature space [12] and learning to transfer object affordance [11]. Finally, Zhang et al. [28] proposed to fuse features of different modalities within a graphical model to produce more discriminative features. We make use of this modality fusion in our transformer model and show that it leads to significant improvements.

3. Unary–pairwise transformers

To leverage the success of transformer-based detectors, we use DETR [1] as our backbone object detector and focus on designing an effective and efficient interaction head for HOI detection, as shown in Fig. 3. The interaction head consists of two types of transformer encoder layers, with the first layer modified to accommodate additional pairwise input. The first layer operates on unary tokens, i.e., individual human and object instances, while the second layer operates on pairwise tokens, i.e., human–object pairs. Based on our analysis and experimental observations in Sec. 4.3 and Sec. 4.4, self-attention in the unary layer preferentially increases the interaction scores for positive HOI pairs, whereas self-attention in the pairwise layer decreases the scores for negative pairs. As such, we refer to these layers as *cooperative* and *competitive* layers respectively.

3.1. Cooperative layer

A standard transformer encoder layer takes as input a set of tokens and performs self-attention. Positional encodings are usually indispensable to compensate for the lack of order in the token set. Typically, sinusoidal functions of the position [27] or learnable embeddings [1] are used for this purpose. It is possible to extend sinusoidal encodings to bounding box coordinates, however, our unary tokens already contain positional information, since they were decoded into bounding boxes. Instead, we take this as an opportunity to inject pairwise spatial information into the transformer, something that has been shown to be helpful

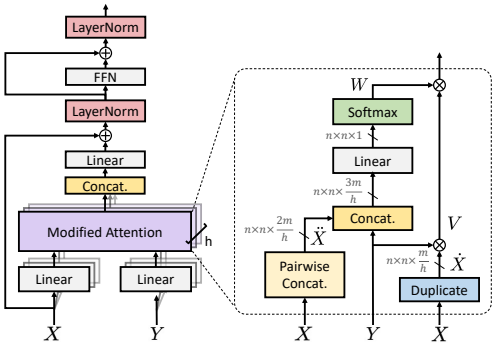


Figure 4. Architecture of the modified transformer encoder layer (left) and its attention module (right). FFN stands for feedforward network [27]. "Pairwise concat." refers to the operation of pairing up all tokens and concatenating the features. "Duplicate" refers to the operation of repeating the features along a new dimension.

for the task of HOI detection [28]. Specifically, we compute the unary and pairwise spatial features used by Zhang et al. [28] from the bounding boxes, including the unary box centre, width, and height, and pairwise intersection-over-union, relative area, and direction, and pass this through an MLP to obtain the pairwise positional encodings. We defer the full details to Appendix A. We also found that the usual additive approach did not perform as well for our positional encodings. So we slightly modified the attention operation in the transformer encoder layer to allow directly injecting the pairwise positional encodings into the computation of values and attention weights.

More formally, given the detections returned by DETR, we first apply non-maximum suppression and thresholding. This leaves a smaller set $\{d_i\}_{i=1}^n$, where a detection $d_i = (\mathbf{b}_i, s_i, c_i, \mathbf{x}_i)$ consists of the box coordinates $\mathbf{b}_i \in \mathbb{R}^4$, the confidence score $s_i \in [0, 1]$, the object class $c_i \in \mathcal{K}$ for a set of object categories $\mathcal{K}$, and the object query or feature $\mathbf{x}_i \in \mathbb{R}^m$. We compute the pairwise box positional encodings $\{\mathbf{y}_{i,j} \in \mathbb{R}^m\}_{i,j=1}^n$ as outlined above. We denote the collection of unary tokens by $X \in \mathbb{R}^{n\times m}$ and the pairwise positional encodings by $Y \in \mathbb{R}^{n\times n\times m}$. The complete structure of the modified transformer encoder layer is shown in Fig. 4. For brevity of exposition, let us assume that the number of heads h is 1, and define

$$\dot{X} \in \mathbb{R}^{n\times n\times m},\ \dot{X}_i \triangleq X \in \mathbb{R}^{n\times m}, \tag{1}$$

$$\ddot{X} \in \mathbb{R}^{n\times n\times 2m},\ \ddot{\mathbf{x}}_{i,j} \triangleq \mathbf{x}_i \oplus \mathbf{x}_j \in \mathbb{R}^{2m}, \tag{2}$$

where $\oplus$ denotes vector concatenation. That is, the tensors $\dot{X}$ and $\ddot{X}$ are the results of duplication and pairwise concatenation. The equivalent values and attention weights can then be computed as

$$V = \dot{X} \otimes Y, \tag{3}$$

$$W = \text{softmax}((\ddot{X} \oplus Y)\mathbf{w} + b), \tag{4}$$

where $\otimes$ denotes elementwise product and $\mathbf{w} \in \mathbb{R}^{3m}$ and $b \in \mathbb{R}$ are the parameters of the linear layer. The output of the attention layer is then computed as $W \otimes V$. Additional details can be found in Appendix D.

3.2. Competitive layer

To compute the set of pairwise tokens, we form all pairs of distinct unary tokens and remove those where the first token is not human, as object–object pairs are beyond the scope of HOI detection. We denote the resulting set as $\{p_k = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{y}_{i,j}) \mid i \neq j, c_i = \text{"human"}\}$. We then compute the pairwise tokens from the unary tokens and positional encodings via multi-branch fusion (MBF) [28] as

$$\mathbf{z}_k = \text{MBF}(\mathbf{x}_i \oplus \mathbf{x}_j, \mathbf{y}_{i,j}). \tag{5}$$

Specifically, the MBF module fuses two modalities in multiple homogeneous branches and return a unified feature representation. For completeness, full details are provided in Appendix C. Last, the set of pairwise tokens are fed into an additional transformer encoder layer, allowing the network to compare the HOI candidates, before an MLP predicts each HOI pair's action classification logits $\widetilde{\mathbf{s}}$.

3.3. Training and inference

To make full use of the pre-trained object detector, we incorporate the object confidence scores into the final scores of each human–object pair. Denoting the action logits of the k^{th} pair p_k as $\widetilde{\mathbf{s}}_k$, the final scores are computed as

$$\mathbf{s}_k = (s_i)^\lambda \cdot (s_j)^\lambda \cdot \sigma(\widetilde{\mathbf{s}}_k), \tag{6}$$

where $\lambda > 1$ is a constant used during inference to suppress overconfident objects [28] and σ is the sigmoid function. We use focal loss[1] [18] for action classification to counter the imbalance between positive and negative examples. Following previous practice [8, 28], we only compute the loss on valid action classes for each object type, specified by the dataset. During inference, scores for invalid combinations of actions and objects (e.g., *eating a car*) are zeroed out.

4. Experiments

In this section, we first demonstrate that the proposed unary–pairwise transformer achieves state-of-the-art performance on both the HICO-DET [2] and V-COCO [7] datasets, outperforming the next best method by a significant margin. We then provide a thorough analysis on the effects of the cooperative and competitive layers. In particular, we statistically show that the cooperative layer increases the scores of positive examples while the competitive layer

[1] Final scores in Eq. (6) are normalised to the interval $[0, 1]$. In training, we instead recover the scale prior to normalisation and use the corresponding loss with logits for numerical stability. See more details in Appendix B.

Table 2. Comparison of HOI detection performance (mAP×100) on the HICO-DET [2] and V-COCO [7] test sets. The highest result in each section is highlighted in bold. ⋆We omit results reported for a set of HICO-DET fine-tuned bounding boxes [5] that we have been unable to reproduce from the authors' code.

		HICO-DET						V-COCO	
		Default Setting			Known Objects Setting				
Method	**Backbone**	Full	Rare	Non-rare	Full	Rare	Non-rare	AP^{S1}_{role}	AP^{S2}_{role}
HO-RCNN [2]	CaffeNet	7.81	5.37	8.54	10.41	8.94	10.85	-	-
InteractNet [6]	ResNet-50-FPN	9.94	7.16	10.77	-	-	-	40.0	-
GPNN [23]	ResNet-101	13.11	9.34	14.23	-	-	-	44.0	-
TIN [16]	ResNet-50	17.03	13.42	18.11	19.17	15.51	20.26	47.8	54.2
Gupta et al. [8]	ResNet-152	17.18	12.17	18.68	-	-	-	-	-
DRG⋆ [5]	ResNet-50-FPN	19.26	17.74	19.71	23.40	21.75	23.89	51.0	-
VSGNet [26]	ResNet-152	19.80	16.05	20.91	-	-	-	51.8	57.0
DJ-RN [14]	ResNet-50	21.34	18.53	22.18	23.69	20.64	24.60	-	-
PPDM [17]	Hourglass-104	21.94	13.97	24.32	24.81	17.09	27.12	-	-
VCL [10]	ResNet-50	23.63	17.21	25.55	25.98	19.12	28.03	48.3	-
ATL⋆ [11]	ResNet-50	23.81	17.43	27.42	27.38	22.09	28.96	-	-
IDN⋆ [15]	ResNet-50	24.58	20.33	25.86	27.89	23.64	29.16	53.3	60.3
HOTR [13]	ResNet-50	25.10	17.34	27.42	-	-	-	55.2	**64.4**
FCL⋆ [12]	ResNet-50	25.27	20.57	26.67	27.71	22.34	28.93	52.4	-
HOI-Trans [29]	ResNet-101	26.61	19.15	28.84	29.13	20.98	31.57	52.9	-
AS-Net [3]	ResNet-50	28.87	24.25	30.25	31.74	27.07	33.14	53.9	-
SCG⋆ [28]	ResNet-101	29.26	**24.61**	30.65	**32.87**	**27.89**	**34.35**	54.2	60.9
QPIC [25]	ResNet-101	**29.90**	23.92	**31.69**	32.38	26.06	34.27	**58.8**	61.0
Ours (UPT)	ResNet-50	31.66	25.94	33.36	35.05	29.27	36.77	59.0	64.5
Ours (UPT)	ResNet-101	32.31	28.55	33.44	35.65	**31.60**	36.86	60.7	66.2
Ours (UPT)	ResNet-101-DC5	**32.62**	**28.62**	**33.81**	**36.08**	31.41	**37.47**	**61.3**	**67.1**

suppresses those of the negative examples. We then visualise the attention weights for specific images, and show how these behaviours are achieved by the attention mechanism. At inference time, our method with ResNet50 [9] runs at 24 FPS on a single GeForce RTX 3090 device.

Datasets: HICO-DET [2] is a large-scale HOI detection dataset with 37 633 training images, 9 546 test images, 80 object types, 117 actions, and 600 interaction types. The dataset has 117 871 human–object pairs with annotated bounding boxes in the training set and 33 405 in the test set. V-COCO [7] is much smaller in scale, with 2 533 training images, 2 867 validation images, 4 946 test images, and only 24 different actions.

4.1. Implementation details

We fine-tune the DETR model on the HICO-DET and V-COCO datasets prior to training and then freeze its weights. For HICO-DET, we use the publicly accessible DETR models pre-trained on MS COCO [19]. However, for V-COCO, as its test set is contained in the COCO val2017 subset, we pre-train DETR models from scratch on MS COCO, excluding those images in the V-COCO test set. For the interaction head, we filter out detections with scores lower than 0.2, and sample at least 3 and up to 15 humans and objects each, prioritising high scoring ones. For the hidden dimension of the transformer, we use $m = 256$, the same as DETR. Additionally, we set λ to 1 during training and 2.8 during inference. For the hyperparameters used in the focal loss, we use the same values as SCG [28].

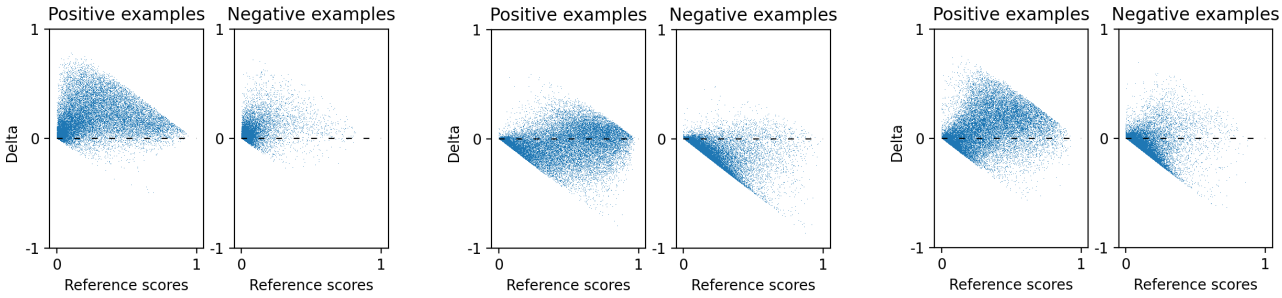
We apply a few data augmentation techniques used in other detectors [1, 25]. Inputs images are scaled such that the shortest side is at least 480 and at most 800 pixels. The longest side is limited at 1333 pixels. Additionally, each image is cropped with a probability of 0.5 to a random rectangle with each side being at least 384 pixels and at most 600 pixels before being scaled. We also apply colour jittering, where the brightness, contrast and saturation values are adjusted by a random factor between 0.6 to 1.4. We use AdamW [22] as the optimiser with an initial learning rate of 10^{-4}. All models are trained for 20 epochs with a learning rate reduction at the 10^{th} epoch by a factor of 10. Training is conducted on 8 GeForce GTX TITAN X devices, with a batch size of 2 per GPU—an effective batch size of 16.

4.2. Comparison with state-of-the-art methods

The performance of our model is compared to existing methods on the HICO-DET [2] and V-COCO [7] datasets

Table 3. Comparing the effect of the cooperative (coop.) and competitive (comp.) layers on the interaction scores. We report the change in the interaction scores as the layer in the Δ Architecture column is added to the reference network, for positives, easy negatives and hard negatives, with the number of examples in parentheses. As indicated by the bold numbers, the cooperative layer significantly increases the scores of positive examples while the competitive layer suppresses hard negative examples. Together, these layers widen the gap between scores of positive and negative examples, improving the detection mAP.

		Δ **Positives** (25 391)		Δ **Easy Negatives** (3 903 416)		Δ **Hard Negatives** (510 991)	
Reference	Δ **Architecture**	Mean	Median	Mean	Median	Mean	Median
Ours w/o coop. layer	+ coop. layer	**+0.1487**	+0.1078	+0.0001	+0.0000	+0.0071	+0.0000
Ours w/o comp. layer	+ comp. layer	-0.0463	-0.0310	-0.0096	-0.0024	**-0.1080**	-0.0922
Ours w/o both layers	+ both layers	**+0.0799**	+0.0390	-0.0076	-0.0018	**-0.0814**	-0.0748



(a) Tab. 3 first row (b) Tab. 3 second row (c) Tab. 3 third row

Figure 5. Change in the interaction score (delta) with respect to the reference score. (a) The distribution of score deltas when adding the cooperative layer (first row of Tab. 3). (b) Adding the competitive layer to the model (second row). (c) Adding both layers (last row). For visualisation purposes, only 20% of the negatives are sampled and displayed.

in Tab. 2. There are two different settings for evaluation on HICO-DET. *Default Setting*: A detected human–object pair is considered matched with a ground truth pair, if the minimum intersection over union (IoU) between the human boxes and object boxes exceeds 0.5. Amongst all matched pairs, the one with the highest score is considered the true positive while others are false positives. Pairs without a matched ground truth are also considered false positives. *Known Objects Setting*: Besides the aforementioned criteria, this setting assumes the set of object types in ground truth pairs are known. Therefore, detected pairs with an object type outside the set are removed automatically, thus reducing the difficulty of the problem. For V-COCO, the average precision (AP) is computed under two scenarios, differentiated by the superscripts $S1$ and $S2$. This is to account for missing objects due to occlusion. For scenario 1, empty object boxes should be predicted in case of occlusion for a detected pair to be considered a match with the corresponding ground truth, while for scenario 2, object boxes are always assumed to be matched in such cases.

We report our model's performance for three different backbone networks. Notably, our model with the lightest-weight backbone already outperforms the next best method by a significant margin in almost every category. This gap is further widened with more powerful backbone networks. In particular, since the backbone CNN and object detection transformer are detached from the computational graph, our model has a small memory footprint. This allows us to use a higher-resolution feature map by removing the stride in the 5^{th} convolutional block (C5) of ResNet [9], which has been shown to improve detection performance on small objects [1]. We denote this as dilated C5 (DC5).

4.3. Macroscopic effects of the interaction head

In this section, we compare the effects of the unary (cooperative) and pairwise (competitive) layers on the HICO-DET test set, with ResNet50 [9] as the CNN backbone. Since the parameters in the object detector are kept frozen for our model, the set of detections processed by the downstream network remains the same, regardless of any architectural changes in the interaction head. This allows us to compare how different variants of our model perform on the same human–object pairs. To this end, we collected the predicted interaction scores for all human–object pairs over the test set and compare how adding certain layers influence them. In Tab. 3, we show some statistics on the change of scores upon an architectural modification. In particular, note that the vast majority of collected pairs are easy negatives with scores close to zero. For analysis, we divide the negative examples into easy and hard, where we define

Table 4. Effect of the cooperative and competitive layers on the HICO-DET test set under the default settings.

Model	Full	Rare	Non-rare
Ours w/o both layers	29.22	23.09	31.05
Ours w/o comp. layer	30.78	24.92	32.53
Ours w/o coop. layer	30.68	24.69	32.47
Ours w/o pairwise pos. enc.	29.98	23.72	31.64
Ours (1× coop., 1× comp.)	31.33	26.02	32.91
Ours (1× coop., 2× comp.)	31.62	**26.18**	33.24
Ours (2× coop., 1× comp.)	**31.66**	25.94	**33.36**

an easy negative as one with a score lower than 0.05 as predicted by the "Ours w/o both layers" model, which accounts for 90% of the negative examples. In addition, we also show the distribution of the change in score with respect to the reference score as scatter plots in Fig. 5. The points are naturally bounded by the half-spaces $0 \leq x + y \leq 1$.

Notably, adding the cooperative layer results in a significant average increase ($+0.15$) in the scores of positive examples, with little effect on the negative examples. This can be seen in Fig. 5a as well, where the score changes for almost all positive examples are larger than zero. In contrast, adding the competitive layer leads to a significant average decrease (-0.11) in the scores of hard negative examples, albeit with a small decrease in the score of positive examples as well. This minor decrease is compensated by the cooperative layer as shown in the last row of Tab. 3. Furthermore, looking at Fig. 5b, we can see a dense mass near the line $y = -x$, which indicates that many negative examples have had their scores suppressed to zero.

Ablation study: In Tab. 4, we ablate the effect of different design decisions on performance. Adding the cooperative and competitive layers individually improves the performance by around 1.5 mAP, while adding both layers jointly improves by over 2 mAP. We also demonstrate the significance of the pairwise position encodings by removing them from the modified encoder and the multi-branch fusion module. This results in a 1.3 mAP decrease. Finally, we observe a slight improvement (0.3 mAP) when adding an additional cooperative or competitive layer, but no further improvements with more layers. As the competitive layer is more costly, we use two cooperative layers.

4.4. Microscopic effects of the interaction head

In this section, we focus on a specific image and visualise the effect of attention in our cooperative and competitive layers. In Fig. 6, we display a detection-annotated image and its associated attention map from the unary (cooperative) layer. The human–object pairs $(1, 4)$, $(2, 5)$ and $(3, 6)$ are engaged in the interaction *riding a horse*. Excluding

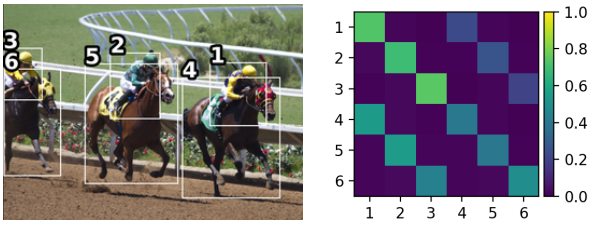


Figure 6. Detected human and object instances (left) and the unary attention map for these instances (right).

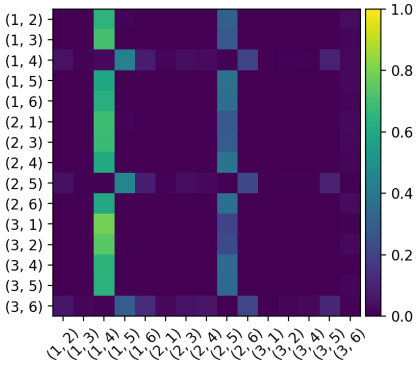
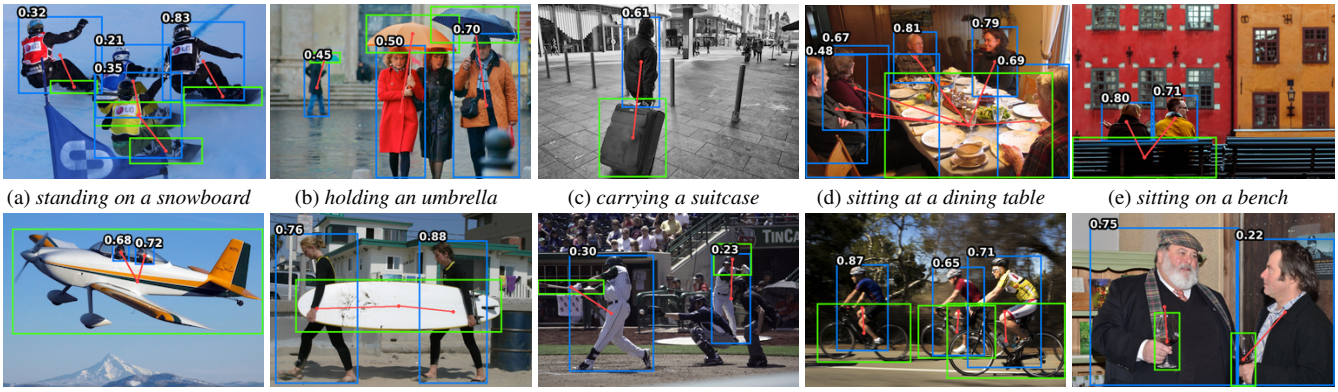


Figure 7. Pairwise attention map for the human and object instances in Fig. 6.

attention weights along the diagonal, we see that the corresponding human and horse instances attend to each other. We hypothesise that attention between pairs of unary tokens (e.g., 1 and 4) helps increase the interaction scores for the corresponding pairs. To validate this hypothesis, we manually set the attention logits between the three positive pairs to minus infinity, thus zeroing out the corresponding attention weights. The effect of this was an average decrease of 0.06 (8%) in the interaction scores for the three pairs, supporting the hypothesis.

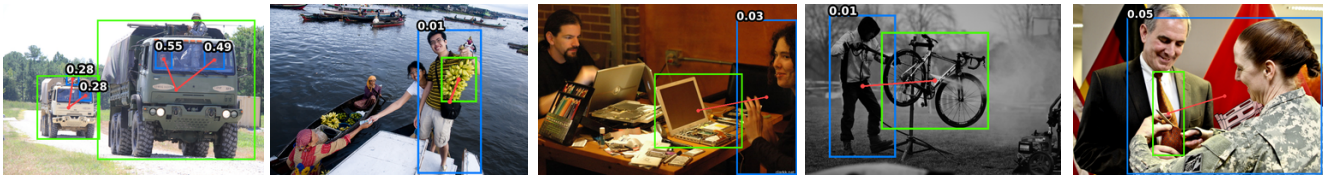
In Fig. 7, we visualise the attention map of the pairwise (competitive) layer. Notably, all human–object pairs attend to the interactive pairs $(1, 4)$, $(2, 5)$ and $(3, 6)$ in decreasing order, except for the interactive pairs themselves. We hypothesise that attention is acting here to have the dominant pairs suppress the other pairs. To investigate, we manually set the weights such that the three interactive pairs all attend to $(1, 4)$ as well, with a weight of 1. This resulted in a decrease of their interaction scores by 0.08 (11%). We then instead zeroed out the attention weights between the rest of the pairs and $(1, 4)$, which resulted in a small increase in the scores of negative pairs. These results together suggest that attention in the competitive layer is acting as a soft version of non-maximum suppression, where pairs less likely to foster interactions attend to, and are suppressed by, the most dominant pairs. More examples and analysis can be found in Appendix E.



(a) *standing on a snowboard* (b) *holding an umbrella* (c) *carrying a suitcase* (d) *sitting at a dining table* (e) *sitting on a bench*

(f) *flying an airplane* (g) *holding a surfboard* (h) *wielding a baseball bat* (i) *riding a bike* (j) *holding a wineglass*

Figure 8. Qualitative results of detected HOIs. Interactive human–object pairs are connected by red lines, with the interaction scores overlaid above the human box. Pairs with scores lower than 0.2 are filtered out.



(a) *driving a truck* (b) *buying bananas* (c) *repairing a laptop* (d) *washing a bicycle* (e) *cutting a tie*

Figure 9. Failure cases often occur when there is ambiguity in the interaction (a), (b), (c) or a lack of training data (c), (d), (e).

4.5. Qualitative results and limitations

In Fig. 8, we present several qualitative examples of successful HOI detections, where our model accurately localises the human and object instances and assigns high scores to the interactive pairs. In particular, for the example in Fig. 8b, our model correctly identifies the subject of an interaction (the lady in red) despite her proximity to a non-interactive human (the lady in black). We also observe in Fig. 8a that our model becomes less confident when there is overlap and occlusion. This stems from the use of object detection scores in our model. Confusion in the object detector often translates to confusion in action classification. We also show five representative failure cases for our model, illustrating its limitations. In Fig. 9a, due to the indefinite position of drivers in the training set (and real life), the model struggled to identify the driver. For Fig. 9d, the model failed to recognise the interaction due to a lack of training data (1 training example), even though the action is well-defined. Overall, ambiguity in the actions and insufficient data are the biggest challenges for HOI detection and our model. Another limitation, specific to our model, is that the computation and memory requirements of our pairwise layer scale quadratically with the number of unary tokens. For scenes involving many interactive humans and objects, this becomes quite costly. Moreover, since the datasets we used are limited and not necessarily representative of images in the wild, we may expect poorer performance on other data, where factors such as the image resolution, lighting quality, and focus may be less controlled.

5. Conclusion

In this paper, we have proposed a two-stage detector of human–object interactions using a novel transformer architecture that exploits both unary and pairwise representations of the human and object instances. Our model not only outperforms the current state-of-the-art—a one-stage detector—but also consumes much less time and memory to train. Through extensive analysis, we demonstrate that attention between unary tokens acts to increase the scores of positive examples, while attention between pairwise tokens acts like non-maximum suppression, reducing the scores of negative examples. We show that these two effects are complementary, and together boost performance significantly.

Potential negative societal impact: Transformer models are large and computationally-expensive, and so have a significant negative environmental impact. To mitigate this, we use pre-trained models and a two-stage architecture, since fine-tuning an existing model requires less resources, as does training a single stage with the other stage fixed. There is also the potential for HOI detection models to be misused, such as for unauthorised surveillance, which disproportionately affects minority and marginalised communities.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, 2020. 2, 3, 5, 6

[2] Yu-Wei Chao, Yunfan Liu, Xieyang Liu, Huayi Zeng, and Jia Deng. Learning to detect human-object interactions. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018. 1, 2, 4, 5, 10, 12, 13

[3] Mingfei Chen, Yue Liao, Si Liu, Zhiyuan Chen, Fei Wang, and Chen Qian. Reformulating hoi detection as adaptive set prediction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2021. 2

[5] Chen Gao, Jiarui Xu, Yuliang Zou, and Jia-Bin Huang. DRG: Dual relation graph for human-object interaction detection. In *Eur. Conf. Comput. Vis.*, 2020. 5

[6] Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. Detecting and recognizing human-object interactions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 5

[7] Saurabh Gupta and Jitendra Malik. Visual semantic role labeling. *arXiv preprint arXiv:1505.04474*, 2015. 4, 5, 12, 13

[8] Tanmay Gupta, Alexander Schwing, and Derek Hoiem. No-frills human-object interaction detection: Factorization, layout encodings, and training techniques. In *Int. Conf. Comput. Vis.*, 2019. 1, 4, 5

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 5, 6, 10

[10] Zhi Hou, Xiaojiang Peng, Yu Qiao, and Dacheng Tao. Visual compositional learning for human-object interaction detection. In *Eur. Conf. Comput. Vis.*, 2020. 5

[11] Zhi Hou, Baosheng Yu, Yu Qiao, Xiaojiang Peng, and Dacheng Tao. Affordance transfer learning for human-object interaction detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

[12] Zhi Hou, Baosheng Yu, Yu Qiao, Xiaojiang Peng, and Dacheng Tao. Detecting human-object interaction via fabricated compositional learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

[13] Bumsoo Kim, Junhyun Lee, Jaewoo Kang, Eun-Sol Kim, and Hyunwoo J. Kim. Hotr: End-to-end human-object interaction detection with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

[14] Yong-Lu Li, Xinpeng Liu, Han Lu, Shiyi Wang, Junqi Liu, Jiefeng Li, and Cewu Lu. Detailed 2d-3d joint representation for human-object interaction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 5

[15] Yong-Lu Li, Xinpeng Liu, Xiaoqian Wu, Yizhuo Li, and Cewu Lu. Hoi analysis: Integrating and decomposing human-object interaction. In *Adv. Neural Inform. Process. Syst.*, 2020. 3, 5

[16] Yong-Lu Li, Siyuan Zhou, Xijie Huang, Liang Xu, Ze Ma, Hao-Shu Fang, Yanfeng Wang, and Cewu Lu. Transferable interactiveness knowledge for human-object interaction detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1, 5

[17] Yue Liao, Si Liu, Fei Wang, Yanjie Chen, Chen Qian, and Jiashi Feng. PPDM: Parallel point detection and matching for real-time human-object interaction detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2, 5

[18] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Int. Conf. Comput. Vis.*, 2017. 1, 4

[19] 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 *Eur. Conf. Comput. Vis.*, 2014. 5, 12

[20] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020. 2

[21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, pages 10012–10022, October 2021. 3

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Int. Conf. Learn. Represent.*, 2018. 5

[23] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *Eur. Conf. Comput. Vis.*, 2018. 1, 5

[24] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Adv. Neural Inform. Process. Syst.*, pages 91–99, 2015. 1, 2

[25] Masato Tamura, Hiroki Ohashi, and Tomoaki Yoshinaga. QPIC: Query-based pairwise human-object interaction detection with image-wide contextual information. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2, 3, 5

[26] Oytun Ulutan, A S M Iftekhar, and B. S. Manjunath. VSGNet: Spatial attention network for detecting human object interactions using graph convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 5

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, volume 30, 2017. 2, 3, 4, 14

[28] Frederic Z. Zhang, Dylan Campbell, and Stephen Gould. Spatially conditioned graphs for detecting human-object interactions. In *Int. Conf. Comput. Vis.*, pages 13319–13327, October 2021. 2, 3, 4, 5, 10

[29] Cheng Zou, Bohan Wang, Yue Hu, Junqi Liu, Qian Wu, Yu Zhao, Boxun Li, Chenguang Zhang, Chi Zhang, Yichen Wei, and Jian Sun. End-to-end human object interaction detection with hoi transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

A. Pairwise positional encodings

We describe the details of the pairwise positional encodings as introduced in Sec. 3.1 of the main paper. Formally, denote a bounding box as $\mathbf{b} = [x, y, w, h]^T \in [0, 1]^4$, where x and y represent the centre coordinates of the bounding box while w and h represent the width and height. Note that these values have been normalised by the image dimensions. For a pair of bounding boxes $\mathbf{b}_1$ and $\mathbf{b}_2$, we start by encoding the unary terms besides the box representation itself, including box areas and aspect ratios as below

$$\mathbf{u} = \mathbf{b}_1 \oplus \mathbf{b}_2 \oplus \left[w_1 h_1, w_2 h_2, \frac{w_1}{h_1}, \frac{w_2}{h_2} \right]^T, \tag{7}$$

where $\oplus$ denotes vector concatenation. We then proceed to encode the pairwise terms as follows

$$\mathbf{p} = \left[\frac{w_1 h_1}{w_2 h_2}, \texttt{IoU}(\mathbf{b}_1, \mathbf{b}_2) \right]^T \oplus f(d_x) \oplus f(d_y), \tag{8}$$

$$d_x = \frac{x_1 - x_2}{w_1}, d_y = \frac{y_1 - y_2}{h_1}, \tag{9}$$

$$f(d) = \left[\texttt{ReLU}(d), \texttt{ReLU}(-d) \right]^T. \tag{10}$$

This includes additional features such as the ratio of box areas, intersection over union (IoU) and directional encodings that characterise the distance between box centres. Note that the directional encodings are normalised by the dimension of the first (human) bounding box instead of that of the image. In addition, function $f(\cdot)$ ensures the componentwise positivity of the feature vector. Finally, denote a multi-layer perceptron as MLP, the complete pairwise positional encoding is computed as below

$$\mathbf{y} = \text{MLP}(\mathbf{u} \oplus \mathbf{p} \oplus \log(\mathbf{u} \oplus \mathbf{p} + \epsilon)), \tag{11}$$

where ϵ is a small constant added to the vector to avoid taking the logarithm of zero.

B. Numerical stability in the loss function

For the sake of numerical stability, loss function for logits is often preferred to that for normalised scores. In our case, due to the fact that the final interaction score is the product of multiple factors, we cannot directly use the loss function for logits. Therefore, we first need to recover the scale prior to normalisation. Denote the normalised object confidence score and action logit as $\hat{y}_1 \in [0, 1]$ and $\hat{y}_2 \in \mathbb{R}$ respectively, the final score is computed as $\hat{y} = \hat{y}_1 \cdot \sigma(\hat{y}_2)$, where σ denotes the sigmoid function. We can then retrieve the corresponding logit $\tilde{y}$ as below

$$\tilde{y} = \sigma^{-1}(\hat{y}), \tag{12}$$

$$= \log \left(\frac{\hat{y}_1}{1 + \exp(-\hat{y}_2) - \hat{y}_1} + \epsilon \right), \tag{13}$$

where ϵ is a small constant added to the term to avoid taking the logarithm of zero.

Table 5. Performance comparison amongst different variants of the cooperative layer on HICO-DET [2] test set under default setting. All variants below use ResNet50 [9] as the backbone CNN and employ one layer. The acronym M.E. stands for modified encoder.

Variant	Full	Rare	Non-rare
Vanilla	31.15 ± .03	25.70	32.77
Vanilla w/ add. pos. enc.	31.14	25.59	32.80
M.E. w/o pairwise terms	30.93	24.53	32.84
M.E.	**31.33** ± .04	**26.02**	**32.91**

C. Multi-branch fusion

The multi-branch fusion (MBF) module [28] employs multiple homogeneous branches, wherein each branch maps the two input features into a subspace of reduced dimension and performs fusion (elementwise product by default). The resultant feature is then mapped back to the original size. Afterwards, elementwise sum is used to aggregate results across all branches. The reduced representation size in a branch is intentionally configured in a way that renders the total number of parameters independent of the number of branches. For brevity of exposition, let us assume the number of branches is 1, for two input vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the output vector $\mathbf{z} \in \mathbb{R}^n$ is computed as

$$\mathbf{z} = W_3^T \phi \left((W_1^T x + \mathbf{b}_1) \otimes (W_2^T y + \mathbf{b}_2) \right) + \mathbf{b}_3, \tag{14}$$

where $W_1, W_2, W_3 \in \mathbb{R}^{n \times n}$ and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathbb{R}^n$ are parameters of linear layers, ϕ refers to the rectified linear unit (ReLU), and $\otimes$ denotes elementwise product. The implementation for this module in PyTorch is shown in Listing 1.

D. Modified transformer encoder layer

In this section, we compare the performance and interpretability of the modified transformer encoder layer to alternative formulations. To this end, we test multiple variants of the cooperative layer. As shown in Tab. 5, using a vanilla transformer encoder results in a small decrease (0.2 mAP) in performance. This gap persists after applying additive positional encodings learned from the unary box terms shown in Eq. (7). We then demonstrate the importance of the pairwise terms in Eq. (8) by removing them from the positional encodings, which resulted in a 0.4 mAP decrease. Together, these results indicate that the pairwise terms provide useful information for the cooperative layer and a consistent mAP performance boost.

In addition, we show the attention maps in different variants of the cooperative layer in Fig. 10. Notably, our modified encoder (Fig. 10b) accurately infers the correspondence between instances, where the interactive humans and objects attend to each other. This suggests that the pairwise positional encoding instills an inductive bias in the modi-

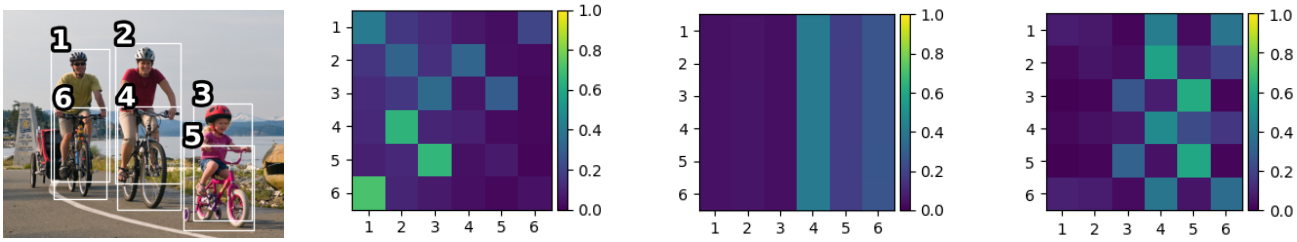
```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class MultiBranchFusion(nn.Module):
    """
    Parameters:
    -----------
    appearance_size: int
        Size of the appearance features
    spatial_size: int
        Size of the spatial features
    hidden_state_size: int
        Size of the intermediate representations
    cardinality: int
        The number of homogeneous branches
    """
    def __init__(self,
        appearance_size: int = 256, spatial_size: int = 256,
        hidden_state_size: int = 256, cardinality: int = 8
    ) -> None:
        super().__init__()
        self.cardinality = cardinality
        sub_repr_size = int(hidden_state_size / cardinality)
        assert sub_repr_size * cardinality == hidden_state_size, \
            "The given representation size should be divisible by cardinality"

        self.fc_1 = nn.ModuleList([
            nn.Linear(appearance_size, sub_repr_size) for _ in range(cardinality)])
        self.fc_2 = nn.ModuleList([
            nn.Linear(spatial_size, sub_repr_size) for _ in range(cardinality)])
        self.fc_3 = nn.ModuleList([
            nn.Linear(sub_repr_size, hidden_state_size) for _ in range(cardinality)])

    def forward(self, appearance: torch.Tensor, spatial: torch.Tensor) -> torch.Tensor:
        return torch.stack([
            fc_3(F.relu(fc_1(appearance) * fc_2(spatial)))
            for fc_1, fc_2, fc_3 in zip(self.fc_1, self.fc_2, self.fc_3)
        ]).sum(dim=0)
```

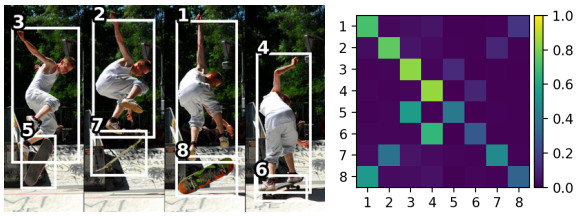
Listing 1. PyTorch implementation of the multi-branch fusion module.



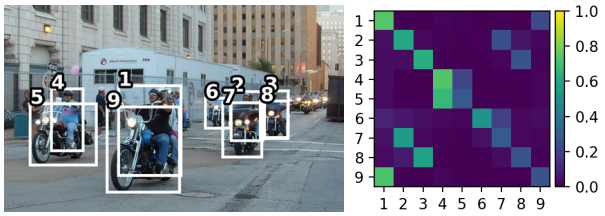
(a) *sample image* (b) *attn. map from M.E.* (c) *attn. map from M.E. w/o pairwise terms* (d) *attn. map from vanilla encoder*

Figure 10. An image with detected instances (a) and attention maps in the cooperative layer with different implementations, including the vanilla encoder (d), modified encoder w/o pairwise terms (c) and the modified encoder (b).

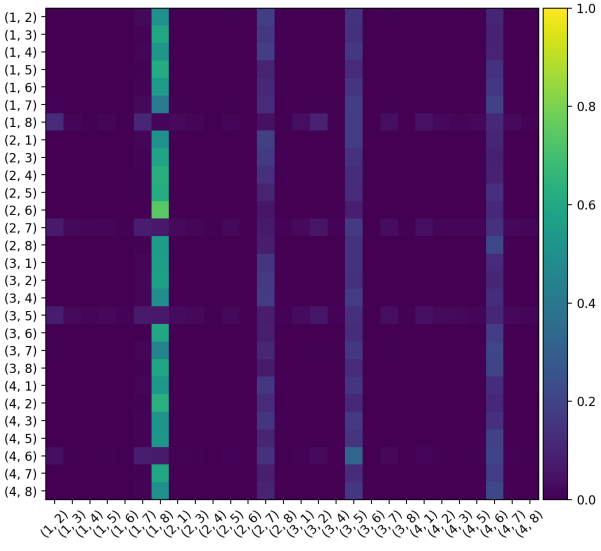
fied encoder that allows it to identify interactive and non-interaction pairs, and preferentially share information between the interactive ones. Furthermore, we show that, without the pairwise terms, as shown in Fig. 10c, the attention map becomes uniform along one dimension. Similarly, the vanilla encoder does not make use of the pairwise spatial information either. This results in the attention maps being much less interpretable as shown in Fig. 10d. In particu-



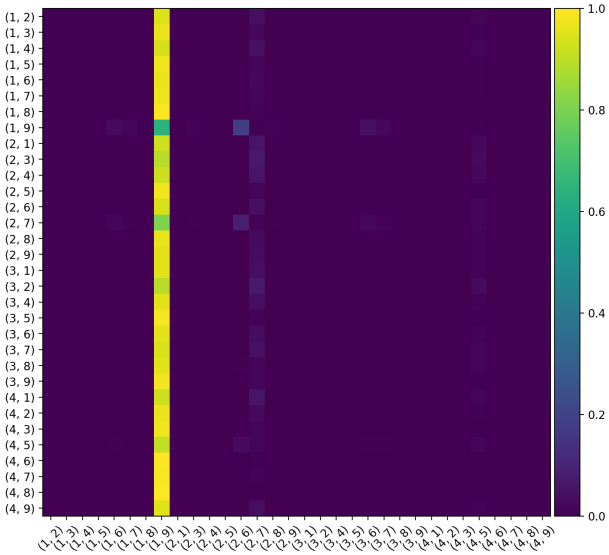
(a) Sample image (left) and the attention map (right) between pairs of unary tokens.



(b) Sample image (left) and the attention map (right) between pairs of unary tokens.



(c) Attention map between pairwise tokens.



(d) Attention map between pairwise tokens.

Figure 11. Our model exhibits the same behaviour on sample images with numerous human and object instances. Specifically, the attention map for unary tokens shows high symmetry, where potentially interactive instances attend to each other. And the pairwise attention map indicates that non-interactive pairs attend to the most dominant pairs to be suppressed.

lar, there is no strong mutual attention between interactive instances, but more attention between non-interactive ones. The complete implementation of the modified encoder in PyTorch is shown in Listing 2.

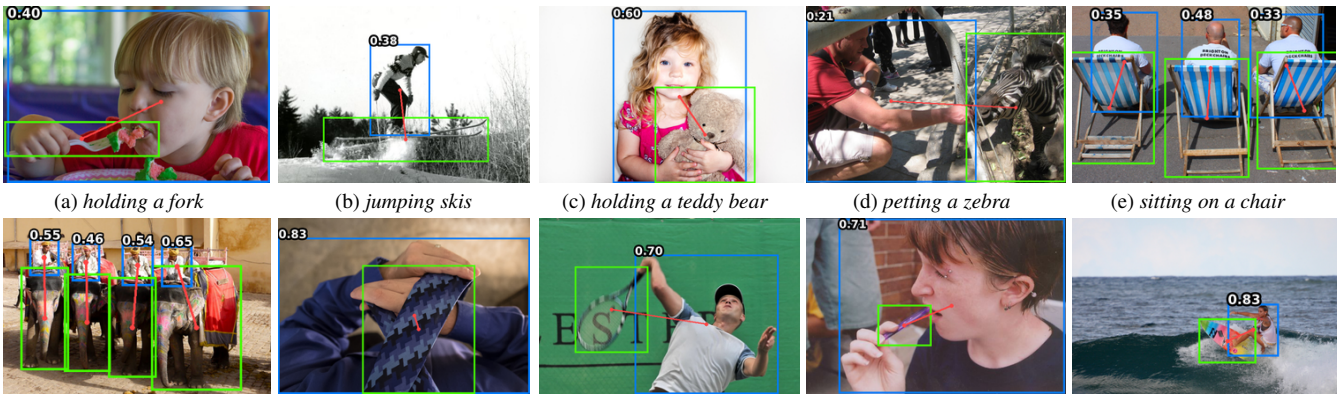
E. Additional qualitative results

We show more visualisations of attention maps in Fig. 11. We intentionally avoided images with very few human and object instances and instead selected those with more complicated scenes for the purpose of demonstration. As shown by the attention maps, our model behaves consistently across different interaction types. More qualitative results for detected HOIs from HICO-DET [2] and V-COCO [7] can be found in Fig. 12 and Fig. 13 respectively.

To better understand the limitations of our model, we also show some false positives on HICO-DET in Fig. 14. In particular, the model sometimes struggles to identify the correct human instance for the interaction as shown in Figs. 14a, 14b and 14d. This is largely due to the plausible spatial relationship and the saliency of the human instance, both of which the model relies on heavily. Another false positive of similar cause can be seen in Fig. 14c, where both human instances are plausible candidates for the interaction *boarding airplane* based on their spatial locations.

F. Asset attrition

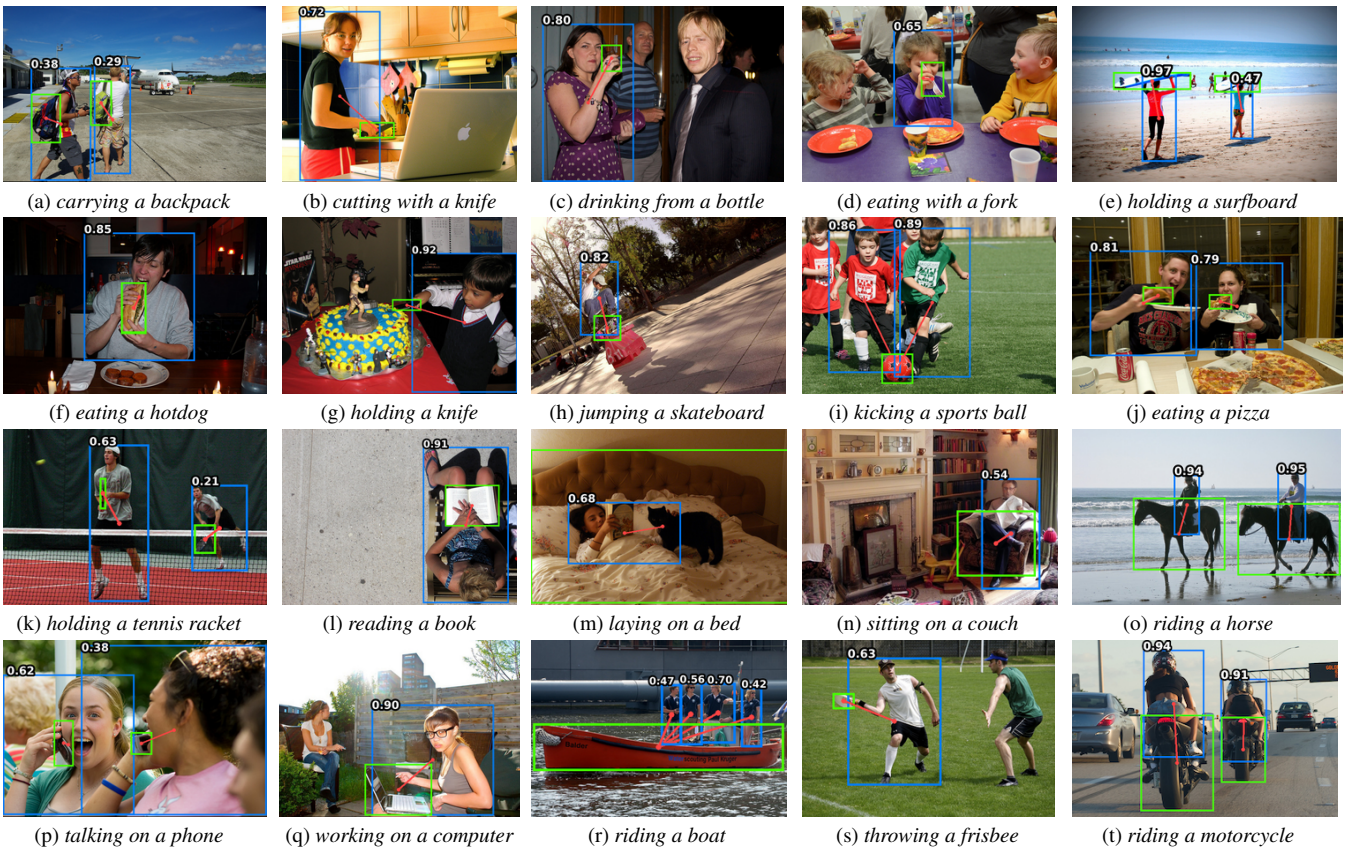
Annotations from the HICO-DET [2] dataset have no license specified. Images from this dataset are licensed under Creative Commons from Flickr. Annotations from the V-COCO [7] dataset are licensed under the MIT License. V-COCO makes use of annotations and images from the MS COCO [19] dataset. The MS-COCO annotations are licensed under a Creative Commons Attribution 4.0 License, and the images are sourced from from Flickr and have a variety of Creative Commons licenses, listed in the MS-COCO annotation files: Attribution-NonCommercial-ShareAlike License, Attribution-NonCommercial License, Attribution-NonCommercial-NoDerivs License, Attribution License, Attribution-ShareAlike License, Attribution-NoDerivs License, and No known copyright restrictions.



(a) *holding a fork* (b) *jumping skis* (c) *holding a teddy bear* (d) *petting a zebra* (e) *sitting on a chair*

(f) *riding an elephant* (g) *wearing a tie* (h) *swinging a tennis racket* (i) *holding a toothbrush* (j) *standing on a surfboard*

Figure 12. Additional qualitative results for detected human–object pairs on HICO-DET [2] test set.



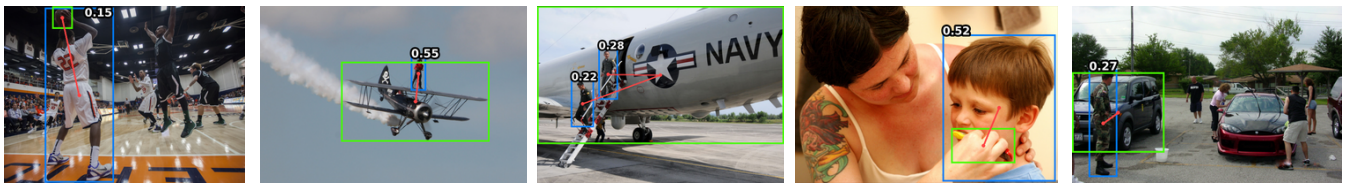
(a) *carrying a backpack* (b) *cutting with a knife* (c) *drinking from a bottle* (d) *eating with a fork* (e) *holding a surfboard*

(f) *eating a hotdog* (g) *holding a knife* (h) *jumping a skateboard* (i) *kicking a sports ball* (j) *eating a pizza*

(k) *holding a tennis racket* (l) *reading a book* (m) *laying on a bed* (n) *sitting on a couch* (o) *riding a horse*

(p) *talking on a phone* (q) *working on a computer* (r) *riding a boat* (s) *throwing a frisbee* (t) *riding a motorcycle*

Figure 13. Additional qualitative results for detected human–object pairs on V-COCO [7] test set.



(a) *blocking a sports ball* (b) *flying an airplane* (c) *boarding an airplane* (d) *brushing with a toothbrush* (e) *washing a car*

Figure 14. False positives on HICO-DET [2] test set.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from typing import Tuple

class ModifiedEncoderLayer(nn.Module):
    def __init__(self, hidden_size: int = 256, repr_size: int = 256, num_heads: int = 8) -> None:
        super().__init__()
        if repr_size % num_heads != 0:
            raise ValueError(
                f"The given representation size {repr_size} "
                f"should be divisible by the number of attention heads {num_heads}.")
        self.sub_repr_size = int(repr_size / num_heads)
        self.hidden_size = hidden_size
        self.repr_size = repr_size
        self.num_heads = num_heads

        self.unary = nn.Linear(hidden_size, repr_size)
        self.pairwise = nn.Linear(repr_size, repr_size)
        self.attn = nn.ModuleList([nn.Linear(3 * self.sub_repr_size, 1) for _ in range(num_heads)])
        self.message = nn.ModuleList([
            nn.Linear(self.sub_repr_size, self.sub_repr_size) for _ in range(num_heads)])
        self.aggregate = nn.Linear(repr_size, hidden_size)
        self.dropout = nn.Dropout(0.1)
        self.norm = nn.LayerNorm(hidden_size)

    def reshape(self, x: torch.Tensor) -> torch.Tensor:
        new_x_shape = x.size()[:-1] + (self.num_heads, self.sub_repr_size)
        x = x.view(*new_x_shape)

        if len(new_x_shape) == 3: return x.permute(1, 0, 2)
        elif len(new_x_shape) == 4: return x.permute(2, 0, 1, 3)
        else: raise ValueError("Incorrect tensor shape")

    def forward(self, x: torch.Tensor, y: torch.Tensor) -> Tuple[torch.Tensor, torch.Tensor]:
        """
        Parameters:
        -----------
        x: torch.Tensor
            Unary tokens of size (N, K)
        y: torch.Tensor
            Pairwise positional encodings of size (N, N, K)
        """
        device = x.device; n = len(x)
        u = F.relu(self.unary(x))
        p = F.relu(self.pairwise(y))
        # Unary features (H, N, K/H)
        u_r = self.reshape(u)
        # Pairwise features (H, N, N, K/H)
        p_r = self.reshape(p)
        # Generate indices for pairwise concatenation
        i, j = torch.meshgrid(torch.arange(n, device=device), torch.arange(n, device=device))
        # Features used to compute attention (H, N, N, 3K/H)
        attn_features = torch.cat([u_r[:, i], u_r[:, j], p_r], dim=-1)
        # Attention weights (H,) (N, N, 1)
        weights = [F.softmax(l(f), dim=0) for f, l in zip(attn_features, self.attn)]
        # Repeated unary feaures along the third dimension (H, N, N, K/H)
        u_r_repeat = u_r.unsqueeze(dim=2).repeat(1, 1, n, 1)
        messages = [l(f_1 * f_2) for f_1, f_2, l in zip(u_r_repeat, p_r, self.message)]
        aggregated_messages = self.aggregate(F.relu(
            torch.cat([(w * m).sum(dim=0) for w, m in zip(weights, messages)], dim=-1)
        ))
        aggregated_messages = self.dropout(aggregated_messages)
        x = self.norm(x + aggregated_messages)
        return x, weights
```

Listing 2. PyTorch implementation of the modified transformer encoder layer. For simplicity, the feedforward network (FFN) [27] has been omitted, as its architecture and implementation has been publicly available.