

Supplementary material: Revisiting spatio-temporal layouts for compositional action recognition

Gorjan Radevski¹²

gorjan.radevski@esat.kuleuven.be

Marie-Francine Moens²

sien.moens@cs.kuleuven.be

Tinne Tuytelaars¹

tinne.tuytelaars@esat.kuleuven.be

¹ PSI-ESAT,

KU Leuven,

Leuven, Belgium

² LIIR-CS Department,

KU Leuven,

Leuven, Belgium

1 Appearance branch and multimodal fusion

In Section 3.3 in the main paper, we discuss the different appearance models we make use of, as well as the multimodal fusion methods employed to combine the appearance and layout (STLT) features.

1.1 Appearance models

As an appearance model we deem a neural network, typically a convolutional neural network (CNN), that takes as input a sequence of video frames $V = (v_0, v_1, v_2, \dots, v_{T-1})$, where T is the number of video frames we sample from the video (Section 3.3, main paper). In certain cases, i.e., with EF, CAF and CACNF we sample a different number of appearance- and layout-based frames. Each $v_i \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels (in our case $C = 3$, as we deal with RGB frames), and H and W are the video height and width, while $V \in \mathbb{R}^{C \times T \times H \times W}$. Depending on the corresponding appearance model, we sample a different set of video frames, as well as pre-process these frames differently during training and evaluation.

1.1.1 2D Resnet152

With the 2D Resnet152 [1] (abbreviated as R2D-152), for each sampled spatial layout frame we sample its corresponding RGB frame, i.e., the frame from where we take the spatial layout (bounding boxes and object categories). As the R2D-152 model was pre-trained on ImageNet [2], we follow the ImageNet way of pre-processing. To be specific, we firstly rescale each frame so that its shorter side length is 256, and then take a center crop of size 224×224 . Each video frame is subsequently per-channel normalized as per standard practice [3]. We take the output of the R2D-152's penultimate layer, i.e., the layer before the

classifier, yielding a frame embedding of size 2048. We concatenate the embeddings of each frame to form a sequence, yielding a video appearance embedding $\hat{A} \in \mathbb{R}^{T \times 2048}$.

1.1.2 2D Resnet50 backbone from Faster R-CNN

We extract the Resnet50 backbone from a Faster R-CNN [15] trained on the task of COCO [16] object detection. In the main paper we abbreviate this model as R2D-50. With the R2D-50 appearance model, to obtain video appearance information, besides the video frames themselves, we additionally utilize the bounding boxes, i.e., the spatial layouts. Given a video frame and its object detections, we firstly pre-process the video frame with the standard Faster R-CNN pre-processing [15], and obtain RoI Align [17] features from each region of interest (as defined by the object detections). Each region of interest feature is of size $256 \times 7 \times 7$, which we average-pool to $256 \times 3 \times 3$, and then flatten to a vector of size 2304.

1.1.3 3D Resnet50

While the input to R2D-152 and R2D-50 is individual frames, with no temporal connectivity between them, with the 3D Resnet50 (abbreviated as R3D or R3D-50 in the main paper), we exploit the temporal connectivity of the frames. To that end, we use a 3D Resnet50 [18], pre-trained on Kinetics-700 [19], Moments in Time [20] and Stair Actions [21]. The input to R3D is a frame sequence $V \in \mathbb{R}^{C \times T \times H \times W}$. During training, we sample a random contiguous sequence of frames, while during evaluation, we sample a contiguous sequence of frames from the center of the video. Note that the sampling of frames is identical to [18] for accurate state-of-the-art comparisons (Section 4.2, main paper). We first rescale each frame so that the shorter side is of size 128. Then, during training, we take a random crop of 112×112 across all frames, and during inference we take a center crop of 112×112 across all frames (see Section 3 for more details). Finally, we forward-propagate the pre-processed video frames through R3D, yielding a video embedding $\hat{A} \in \mathbb{R}^{2048 \times T_d \times H_d \times W_d}$, where $_d$ indicates downscaled, and 2048 is the R3D penultimate layer hidden size.

1.1.4 3D Resnet50 Transformer

With the 3D Resnet50 Transformer appearance model (R3D-Transformer), we reuse R3D, as defined and explained in Section 1.1.3, and we add a transformer model [19] on top. Namely, given the output video embedding from R3D, we firstly use an additional Conv3D layer with a kernel size $1 \times 1 \times 1$ to project it in the desired hidden dimensionality $\hat{A} \in \mathbb{R}^{Z \times T_d \times H_d \times W_d}$, where Z is the hidden size. We flatten the embedding across the temporal and spatial dimension $\mathbb{R}^{Z \times T_d \times H_d \times W_d}$, essentially making it a sequence, and append a special `class` embedding. We sum the R3D embedding with additional positional embeddings to inject positional information. We forward-propagate the obtained embeddings through the transformer, yielding a sequence of hidden states as output.

1.2 Multimodal fusion

With the multimodal fusion methods, our goal is to feed the appearance information (obtained using the models from Section 1.1) in the Spatial-Temporal Layout Transformer (STLT). As each appearance model yields a unique video embedding \hat{A} , we devise different approaches to inject the video embedding in STLT.

1.2.1 Per-frame fusion (PFF)

With the per-frame fusion (PFF) method (Figure 2 (a), main paper), our goal is to inject the R2D-152 frame embeddings (see Section 1.1.1) in STLT. As we have a separate embedding for each frame, we feed the embeddings directly in STLT’s Temporal Transformer module. Note that the output of the Spatial Transformer module for a single frame \hat{s}_i is summed with a positional embedding $\hat{s}_i + \hat{p}_i$, and is provided as input to the Temporal Transformer module. Here, with PFF, we sum the R2D-152 frame embedding \hat{a}_i^f together with \hat{s}_i and \hat{p}_i , apply layer-normalization [14] and dropout [15]: $\hat{t}_i = \text{Dropout}(\text{LayerNorm}(\hat{s}_i + \hat{p}_i + \hat{a}_i^f))$. Then, as in STLT, we provide \hat{t}_i as input to the Temporal Transformer. The remaining part of the model follows STLT.

1.2.2 Per-box fusion (PBF)

The goal with the per-box fusion (PBF) method (Figure 2 (b), main paper) is to inject appearance information at a low, object-specific level. To that end, we utilize the appearance video features obtained from R2D-50 (see Section 1.1.2). With STLT, given a frame object o_j , we obtain its embedding as $\hat{o}_j = \text{Dropout}(\text{LayerNorm}(\hat{c}_j + \hat{l}_j))$. With PBF, the embedding for a single object region in a single video frame is denoted as \hat{a}_j^r , which we fuse with the object embedding as $\hat{o}_j = \text{Dropout}(\text{LayerNorm}(\hat{c}_j + \hat{l}_j + \hat{a}_j^r))$. Then, same as STLT, we provide the object embeddings \hat{f}_i for a single frame f_i to the Spatial Transformer.

1.2.3 Early fusion (EF)

With the early fusion (EF) method (Figure 2 (c), main paper), we inject the R3D video appearance features into STLT. First, we obtain $\hat{A} \in \mathbb{R}^{2048 \times T_d \times H_d \times W_d}$ from R3D, perform average pooling across both the temporal (T_d) and the spatial dimensions (H_d and W_d), and project the embedding in the corresponding hidden size using a fully-connected layer. Then, we prepend the video appearance embedding \hat{A} as first element in the sequence of layout-based frame representations obtained from the Spatial Transformer. By doing so, each of the layout-based frame representations can attend on the R3D video embedding.

1.2.4 Video Action Transformer Fusion (VATF)

We take inspiration from the Video Action Transformer [16], and attempt to conceptually adapt it to perform multimodal fusion. We abbreviate this fusion method as VATF, and display it in Figure 2 (d) – main paper. To achieve such fusion, given the R3D appearance embedding \hat{A} , we firstly perform average pooling across the time dimension, such that $\hat{A} \in \mathbb{R}^{2048 \times H_d \times W_d}$. Then, given a set of bounding boxes corresponding to the temporally central frame, we perform RoI Align and obtain region of interest embeddings. Finally, the non-temporally pooled R3D video embedding represents the memory, while the regional embeddings represent the query as input to a transformer model. We select the output hidden state corresponding to the bounding box of the whole frame ($[0, 0, w, h]$) and add a classifier on top. For details refer to Video Action Transformer [16].

1.2.5 Late concatenation fusion (LCF)

Late concatenation fusion (LCF) is always used a strong baseline for multimodal fusion. We visualize the method in Figure 2 (e) – main paper. It follows the general paradigm for late fusion, where we obtain the STLT and R3D outputs, linearly project them with separate fully-connected layers so that both embeddings have the same dimensionality, and concatenate the representations before the classifier.

1.2.6 Cross-attention fusion (CAF)

The main building block of Cross-Attention Fusion, abbreviated as CAF, (Figure 2 (c), main paper) is the cross-attention module. We draw inspiration from successful methods for fusion of text and image embeddings [18], mainly used for visual question answering, image-text matching, etc. To be specific, we get STLT’s and R3D-Transformer’s hidden states, $\hat{H} \in \mathbb{R}^{Z \times T}$ and $\hat{A} \in \mathbb{R}^{Z \times T_d * H_d * W_d}$ respectively, where Z is the hidden size. T does not need to be equal to $T_d * H_d * W_d$, i.e., we can increase or decrease the number of sampled layout-based or appearance-based frames. However, they both have to have the same hidden size Z . Then, the module consists of cross-attention, self-attention, and feed-forward module. The cross-attention is applied on each branch separately, such that for the layout branch, the multi-head attention queries are \hat{H} , while the keys and values are \hat{A} . When the cross-attention is applied on the appearance branch, the queries are \hat{A} , while the keys and values are \hat{H} . The remaining modules (self-attention, feed-forward module) closely follow the standard transformer block (refer to [19] for details). Finally, we select the output hidden state corresponding to `class` from both the layout $\hat{h}_{\text{class}}^{\text{caf}}$ and the appearance $\hat{a}_{\text{class}}^{\text{caf}}$ branch, concatenate them, and add a classifier on top.

1.2.7 Cross-attention CentralNet fusion (CACNF)

Our motivation for performing multimodal fusion of the layout and appearance branch representations is their complementarity. But even though CAF follows state-of-the-art practices for fusion of image and text embeddings, we observe that CAF performance is comparable to LCF (Table 1, main paper), due to the layout and appearance branch not preserving their individual capabilities (Figure 3, main paper). To overcome this issue, while keeping the fusion module intact, we use a multimodal fusion method proven to work well across different tasks – CentralNet [20]. To that end, CAF remains as is, however, before forward-propagating \hat{H} and \hat{A} through CAF, we select the hidden states corresponding to `class`, namely \hat{h}_{class} and \hat{a}_{class} , and add classifiers (implemented as linear layers) on top of each of them.

During training, we minimize three cross-entropy losses between the model predictions and the target action: (1) STLT loss, (2) R3D-Transformer loss, (3) CAF loss.

During inference, we average the logits of the three classifiers and take the action with the maximal probability as the model’s prediction.

2 Baseline layout-based models

In Section 4.1 in the main paper, we compare STLT performance on the Something-Else compositional dataset and the Something-Something V2 dataset against (i) GNN-NL: A

baseline model [13] utilizing a graph neural network [14] for spatial reasoning and a non-local neural network [20] for temporal reasoning. Essentially, our implementation of STIN [13], where we train the model within our setup (optimizer, learning rate scheduler, training epochs, etc.). Refer to [13] for details; (ii) S&TLT: An STLT variant, where the spatial and temporal reasoning are performed jointly.

2.1 S&TLT

With the Spatial & Temporal Layout Transformer, abbreviated as S&TLT, instead of having decoupled spatial and temporal reasoning over the spatio-temporal layouts, we perform it jointly. Similar to STLT, we obtain the category embedding \hat{c}_j and the location in the frame embedding \hat{l}_j with two separate fully-connected layers. However, in this case, for each frame object o_j , we additionally obtain its frame position index, e.g., if the object is from the first frame the index would be 0, while if it is from the last frame, it would be $n - 1$, where n is the number of sampled video frames. Therefore, for a single object, we obtain its embedding as $\hat{o}_j = \text{Dropout}(\text{LayerNorm}(\hat{c}_j + \hat{l}_j + \hat{p}_j))$, where an additional fully-connected layer yields the \hat{p}_j embedding. Finally, we append a special `class` object embedding \hat{o}_{class} at the sequence end, and forward-propagate the sequence of object embeddings through a transformer model. We control the attention pattern using an attention mask, such that for objects spanning a single frame we perform bidirectional attention, while we perform causal attention across the video.

3 Experimental setup

For all layout-based models we randomly sample 16 frames represented as spatio-temporal layouts. For models using an R3D appearance model, we uniformly sample 32 frames from the video, and then resize the frames so that the size of the smaller side is 128. During training, we randomly crop the frames to 112×112 and apply colour jittering. During inference, we take a center crop of size 112×112 .

We train all models for 20 epochs with AdamW [21], with a peak learning rate of $5e - 5$, linearly warmed-up for the first 10% of the training, and decreased to 0.0 until the end. We apply weight decay of $1e - 3$, gradient clipping when the norm exceeds 5.0, and dropout [22] of 0.1 in the attention and feed-forward modules (the bias term, the positional and `class` embeddings are excluded from the weight decay). We use 4 and 8 multi-head self-attention layers for the spatial and temporal transformer in STLT respectively, and 4 cross- and self-attention in CAF and CACNF. The hidden size for all modules is 768.

On the Something-Something and Something-Else dataset, only two object categories are registered in STLT – “hand” or “object”. On the Action Genome dataset, we register all 36 object categories present in the dataset (also predicted by the trained object detector) – “hair”, “book”, “medicine”, “vacuum”, “food”, “groceries”, “floor”, “mirror”, “cabinet”, “doorway”, “notebook”, “picture”, “phone”, “couch”, “sandwich”, “bottle”, “towel”, “box”, “blanket”, “television”, “bag”, “refrigerator”, “table”, “light”, “broom”, “shoe”, “doorknob”, “bed”, “window”, “shelf”, “door”, “pillow”, “laptop”, “dish”, “clothes” and “person”.

All models trained on the Something-Something and the Something-Else datasets minimize the cross-entropy loss, while all models trained on the Action Genome dataset minimize the per-class binary cross-entropy loss during training.

4 Datasets details and statistics

In Table 1, for each dataset used we show the classification problem type, the number of action classes, as well as the number of training and validation videos.

Dataset	Classification problem type	Number of actions	Number of training videos	Number of validation videos
Something-Else: Compositional dataset [16]	Multi-class classification	174	54,919	57,876
Something-Else: few-shot pre-training dataset [16]	Multi-class classification	88	112,397	12,467
Something-Else: 5-shot dataset [16]	Multi-class classification	86	430	49,822
Something-Else: 10-shot dataset [16]	Multi-class classification	86	860	43,954
Something-Something dataset [17]	Multi-class classification	174	168,913	24,777
Action Genome dataset [8]	Multi-class and multi-label classification	157	7,787	1,814

Table 1: Statistics of each of the datasets used in the paper.

Next, we provide the remaining details for each of the datasets as well as an explanation of the setup in which the experiments are conducted.

4.1 Something-Else: Compositional dataset

In the Something-Else [16] compositional split, the videos are divided in a training and validation set based on the objects present in each video. Firstly, all frequent object categories are found, i.e., object categories that appear more than 100 times in the videos, and those videos are kept. Secondly, the videos that contain the frequent object categories are split in two disjoint sets. In this setup, the models encounter the same action classes during training and evaluation, however, the objects present in the videos are non-overlapping between training and evaluation, i.e., the models encounter strictly novel objects during evaluation.

4.2 Something-Else: Few-shot dataset

In the few-shot version of the Something-Else dataset, the goal is validate to what extent the models can learn to generalize in a low-data regime. To that end, the videos are split in two sets: (1) A base set – used for pre-training the models, containing 88 action classes out of the 174 in total; (2) A few-shot set – used for fine-tuning the models, containing the remaining 86 actions classes. The few-shot split has two variants, a 5-shot and 10-shot split, where in each there are 5 and 10 videos for each action class respectively. Besides the need to generalize from few samples, in the few-shot split the constraint of compositionality still holds, i.e., the objects that appear during training do not overlap with the ones during validation.

For the experiments on the Something-Else few-shot dataset, we follow the exact setup of [16] for proper comparisons w.r.t. state-of-the-art (Table 2, main paper) – we pre-train the models on the base set of videos, then freeze the model weights except the classifier which is trained on the few-shot split.

5 Action Genome: Object detector training

The Action Genome dataset [8], built on top of Charades [16], provides bounding box and object category annotations for a random subset of frames where the action occurs in each video. Besides utilizing these in the “oracle” experiments, we train a Faster R-CNN [18] object detector to create an “object predictions” (obj. predictions) setting. For training the object detector we use the standard train/validation Charades split. Using Detectron2 [22],

we train a Faster R-CNN with a Resnet101 [6] backbone, which was pre-trained on the COCO [11] dataset. We train the model with stochastic gradient descent (SGD) for 5 epochs, with a learning rate of $1e-5$. During training, we apply random horizontal flipping of the frames. We obtain an average precision (AP) of 11.3 on the Action Genome validation set. When generating the spatio-temporal layouts, we keep all object detections with a probability of at least 0.5.

6 Additional qualitative evaluation

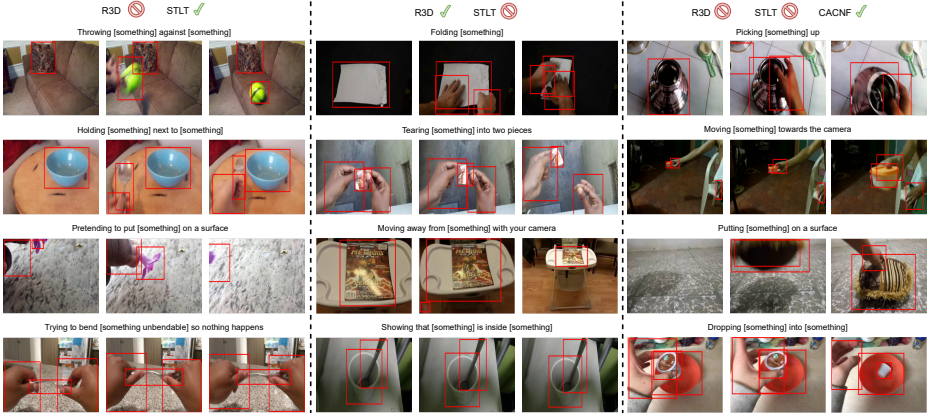


Figure 1: Qualitative evaluation on the Something-Else compositional dataset. **Left:** R3D mispredicts, STLT predicts correctly. **Middle:** STLT mispredicts, R3D predicts correctly. **Right:** STLT and R3D mispredict, CACNF predicts correctly. Above each set of video frames is the ground truth action class.

We provide additional qualitative evaluation on the Something-Else compositional split. We follow the same setup as in Figure 4 in the main paper, where we are interested in visually inspecting three error types: (i) R3D predicts wrong, STLT predicts correct action; (ii) STLT predicts wrong, R3D predicts correct action; (iii) STLT and R3D predict wrong, CACNF predicts correct action. We show the results in Figure 1.

Lastly, we visually inspect how STLT copes with background clutter, and plot two error types: (i) I3D predicts the correct action with probability less than 0.5, STLT predicts the correct action with probability higher than 0.5; (ii) STLT predicts the correct action with probability less than 0.5, I3D predicts the correct action with probability higher than 0.5. Note that in both scenarios, we show the frames where the action of interest occurred, i.e., the action for which there is a mismatch between the models' predictions. Note that there may be other actions occurring simultaneously, or in other parts of the video. The results are shown in Figure 2.



Figure 2: Qualitative evaluation on the Action Genome. **Left:** I3D mispredicts, STLT predicts correctly. **Right:** STLT mispredicts, I3D predicts correctly. Above each set of 3 video frames is the ground truth action class.

References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.
- [3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [4] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.
- [5] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, 2017.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [8] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.
- [9] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3d cnns? *arXiv preprint arXiv:2004.04968*, 2020.
- [10] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [11] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.
- [12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.

- [13] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1049–1059, 2020.
- [14] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.
- [15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL <https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf>.
- [16] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.
- [17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [18] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- [19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.
- [20] Valentin Vielzeuf, Alexis Lechervy, Stéphane Pateux, and Frédéric Jurie. Centralnet: a multilayer approach for multimodal fusion. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.
- [21] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.
- [22] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [23] Yuya Yoshikawa, Jiaqing Lin, and Akikazu Takeuchi. Stair actions: A video dataset of everyday home actions. *arXiv preprint arXiv:1804.04326*, 2018.