Supplementary Materials for Unsupervised Open-Vocabulary Object Localization in Videos

1. Datasets and metrics

1.1. Datasets

We utilize Something-Something-v2 [3] to pre-train our vision backbone and use ImageNet-1K [2] to fine-tune the patch-based CLIP model. To train and evaluate video slots learning and labeling, we opt for two datasets, ImageNet-VID [7] and YouTube-VIS [8], which focus on video objects.

Something-Something-v2 is a vast video collection showcasing individuals performing actions in specific environments. Something-Something-v2 contains around 169k training videos and 25k validation videos from 174 classes. Each video features both an action and object, and merely considering appearance information would not suffice. Thus, we employ the video files to pre-train our VideoMAE, disregarding any annotations.

ImageNet-VID is a commonly used benchmark for video object detection. The dataset comprises 3862 and 555 video snippets in its training and validation sets, respectively. The video streams are annotated on every frame at either 25 or 30 fps. Furthermore, it includes 30 object categories, which are a subset of the categories in the ImageNet DET dataset.

YouTube-VIS is a large-scale video instance segmentation dataset, including 2,883 high-resolution YouTube videos with pixel-level annotations of object instances. The videos are diverse in terms of scene complexity, motion, and object categories, making it a challenging dataset for video object tasks. Although initially designed for instance segmentation, YouTube-VIS includes object bounding boxes, making it compatible with video object detection. Specifically, since the annotation of YouTube-VIS validation set and test set is inaccessible, we make a customized train-val split randomly.

1.2. Metrics

Detailed definition of type of slots (SO/PO/GO/BG) The part-whole hierarchies has been a long-existing issue in object-centric learning [4]. As for a visually complex object, the localization has the chance to split it in multiple slots. We categorize slots that overlap with objects as containing a Single Object (SO), Part of an Object (PO) or a Group of Objects (GO). We utilize intersection, area, and union of the slots and objects to determine their relations and help categorizing slots.

Formally, for each predicted bounding box b_p derived from one slot and all the ground truth bounding box b_g, we have the following rules to categorize this slot with respect to two threshold parameters τ_1 and τ_2:

1. If the intersection over union (IoU) between the two boxes are larger than τ_1, we regard it as Single Object. If there exists only one b_g such that the intersection over b_g's area is larger than τ_2, we also regard it as Single Object.

2. Else, if the intersection over b_p's area is larger than τ_2, we regard it as Part of an Object.

3. Else, if there exists multiple b_g such that the intersection over each b_g's area is larger than τ_2, we regard it as Group of Objects.

4. Finally, if none of the above conditions hold, the slot should be Back-Ground slot.

In practice, we set $\tau_1 = 0.5$ and $\tau_2 = 0.5$. Based on the definition, we propose to use the proportion of each type of slots as a metric to evaluate how well the object-centric model can handle the part-whole problem.

2. Implementation details

2.1. Training of vision backbone

We conduct the experiments with 128 NVIDIA Tesla T4 GPUs for pre-training of VideoMAE on Something-Something V2 datasets. Specifically, we use ViT-Base as our encoder and a four-layer transformer as our decoder. The patch size is set to $16 \times 16 \times 1$. The videos are sampled to length of 16 and cropped to 224×224. The videos are reconstructed given the 90% patches masked.

For optimizer, we utilize AdamW [6] with a base learning rate of $1.5e-4$. We pre-train the model for 800 epochs with 40 warm-up epochs and cosine learning rate scheduler. The total batch size is 512.

2.2. Training of video slot learning

We train the video slot learning module using the Adam optimizer [5] with a learning rate of $3e^{-3}$, linear learning rate warm-up of 5 000 optimization steps and an exponentially decaying learning rate schedule. Further, we clip the gradient norm at 1 in order to stabilize training. The experiments on ImageNet-VID and YouTube-VIS are conducted separately. We train the model on specific dataset for 300 epochs. The models were trained on 8 NVIDIA Tesla T4 GPUs with a local batch size of 10, where each sample contains 8 frames. The number of slots is set to 15.

Training with variable resolution There are some special designs to handle variable resolution in these datasets. For the data, we first resize the short side of the video into 224 while keeping the aspect ratio. Then we resize its long side to the nearest multiple of patch size to make it fit the patchify operation in the ViT model. For different models involved in the pipeline, we employ positional embedding interpolation techniques in both vision backbone (VideoMAE) and patch-based CLIP. As for the video slot grouping module, slot attention is naturally feasible to handle variable length tokens. We make similar positional embedding interpolation in the slot decoder part. During training, we random sample frames repeatedly from one video for each batch to avoid the issue of batching variable length tokens.

2.3. The finetuning of patch-based CLIP

For patch-based CLIP, we finetune a pre-trained ViT-B/16 CLIP model on ImageNet 1K dataset, which includes 1.28M images. For each image, we augment the training with a random resized crop with scale between 0.9 and 1 and resize the cropped area to 224× 224, then a random horizontal flip. The model is finetuned for 200 epochs on 32 NVIDIA Tesla T4 GPUs with a local batch size of 4096. The optimizer is SGD, and the learning rate scheduler is cosine annealing with initial learning rate of 1.

2.4. Inference pipeline

In this part, we give a detailed explanation of the full inference pipeline. Given a video $V \in \mathbb{R}^{T\times H\times W\times 3}$ as input, we explicitly divide it into three steps.

(1) Video Slot Extraction. We first patchify the video and extract its spatiotemporal patch features with shape $\mathbb{R}^{T\times H'\times W'\times D}$ by the vision backbone, where T is number of frames, H and W are the height and width of the video, H' and W' are the height and width after patchify operation, D is the hidden dimension. Next, we feed the patch features to the trained spatiotemporal grouping module to get the patch-slot assignment $\alpha \in \mathbb{R}^{K\times T\times H'\times W'}$, where K is the number of slots. The patch-slot assignment serves as the preliminary localization.

(2) Slot Labeling. We feed the same video V to the patch-based CLIP frame by frame to get the semantic patch features with shape $\mathbb{R}^{T\times H'\times W'\times D}$. Next, the patch-slot assignment α is employed to aggregate the semantic patch features into semantic slots features with shape $\mathbb{R}^{K\times D}$ by average pooling. Then we use the semantic slots features to match a set of text features with shape $\mathbb{R}^{Q\times D}$ extracted from text prompts, where Q is the number of text prompts. Next, we take the top probability as the semantic label of the slot and get the named localization.

(3) Joint Optimization. Given the named localization from last step, we further process them with the joint optimization. As described in the main paper, the joint optimization is mainly responsible for excluding some slots of not interests and merge some neighbor slots with the same semantic label.

3. Qualitative results

3.1. Failure cases analysis

We visualize and discuss four typical failures or mistakes by our model.

Localization failure This type of failure is mainly caused by inaccurate segmentation of our models. In each samples in Fig. 1, since some characteristic features of the object are included in one slot that overlaps partially with the object, *e.g.* the ear of the dog or the dorsal fin of the whale, our model still recognizes their name, and that leads to inaccurate localization but correct naming.

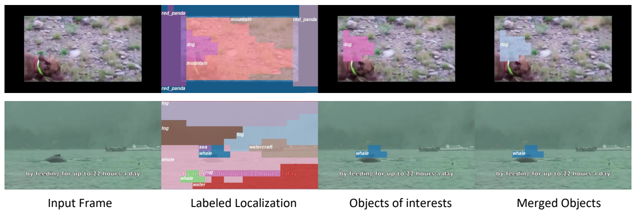

Figure 1. Examples of localization failure.

Background removal mistakes Our model removes two types of background slots in post-processing: the slots named with common background labels and the slots resembling neither target nor common background labels. However, the pipeline may wrongly remove the object of interest. For example, in Fig. 2, the car and the train are over-segmented into several parts. Although our patch-based CLIP can recognize the class of the patch, however, the cosine similarity of such slots with the text vector may be small. In such circumstances, those parts of the target object may be removed, leading to incomplete objects after merging.

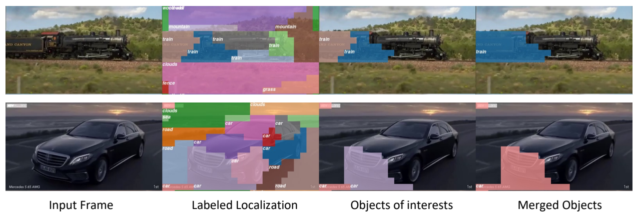

Figure 2. Examples of background removal mistakes.

Multiple instance failure Limited by the insensitivity of CLIP model to quantity words, we cannot distinguish how many objects (especially objects of the same type) in a slot. The merging process tends to merge objects with the same label into groups of objects. For example, in Fig. 3, bicycles and antelopes are merged together respectively in each sample due to the shared semantic meaning and close spatial relationship across multiple instances.

Moreover, due to the resolution limit of slot attention mechanism, our model cannot always segment adjacent small objects. For example, the small airplanes are grouped together by the slot attention in the last row of Fig. 3.

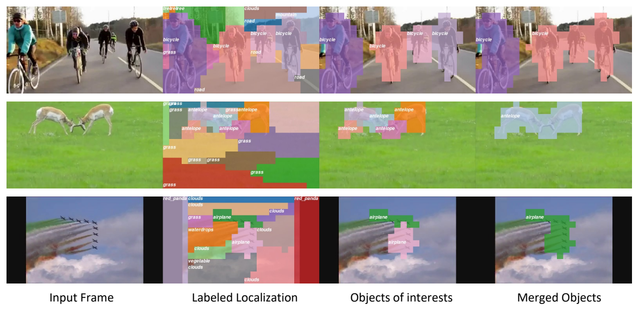

Figure 3. Examples of multiple instance failure.

Naming mistake The patch-based CLIP may not be able to recognize every slot. On one hand, the word may have multiple meanings, in the first row Fig. 4, the group of people is named as “cattle”, due to cattle also means “human beings especially en masse”. On the other hand, the patch-based CLIP may not be able to recognize the slot if the slot is just a part of an object. In the second row Fig. 4, the hair of cat is recognized as lion, which is also reasonable since the hair may also come from a lion.

3.2. More qualitative examples

We show more qualitative examples on ImageNet-VID and YouTube-VIS datasets in Figure 5 and Figure 6, respectively.

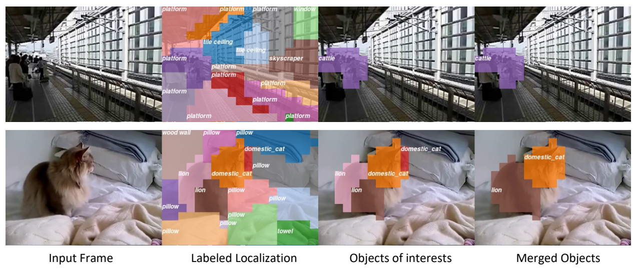

Figure 4. Examples of naming mistakes.

4. Prompt engineering

To fully utilize the language models, we do not directly use the embedding of the words for classes in Youtube-VIS or ImageNet-VID as the text features. But instead, we embed each class name into the sentence with its form as “a photo of a [CLASS]”. For *common background labels*, we select the stuff classes of COCO-stuff [1]. Generally, stuff classes correspond to amorphous background regions such as grass and sky.

Some minor modifications are made to transfer the format of some classes names, e.g. from “wall-concrete” to “concrete wall”. Interestingly, though we add the article “a” in the text template, our model can still match to patches with multiple objects. To some extent, this can further help explain why our model doesn’t distinguish groups of same-category instances together. We also tried adding some synonyms of some classes, for example, “ape” for “monkey”, and “impala” for “antelope”. However, the improvement of synonyms is not significant and we report the results without synonyms.

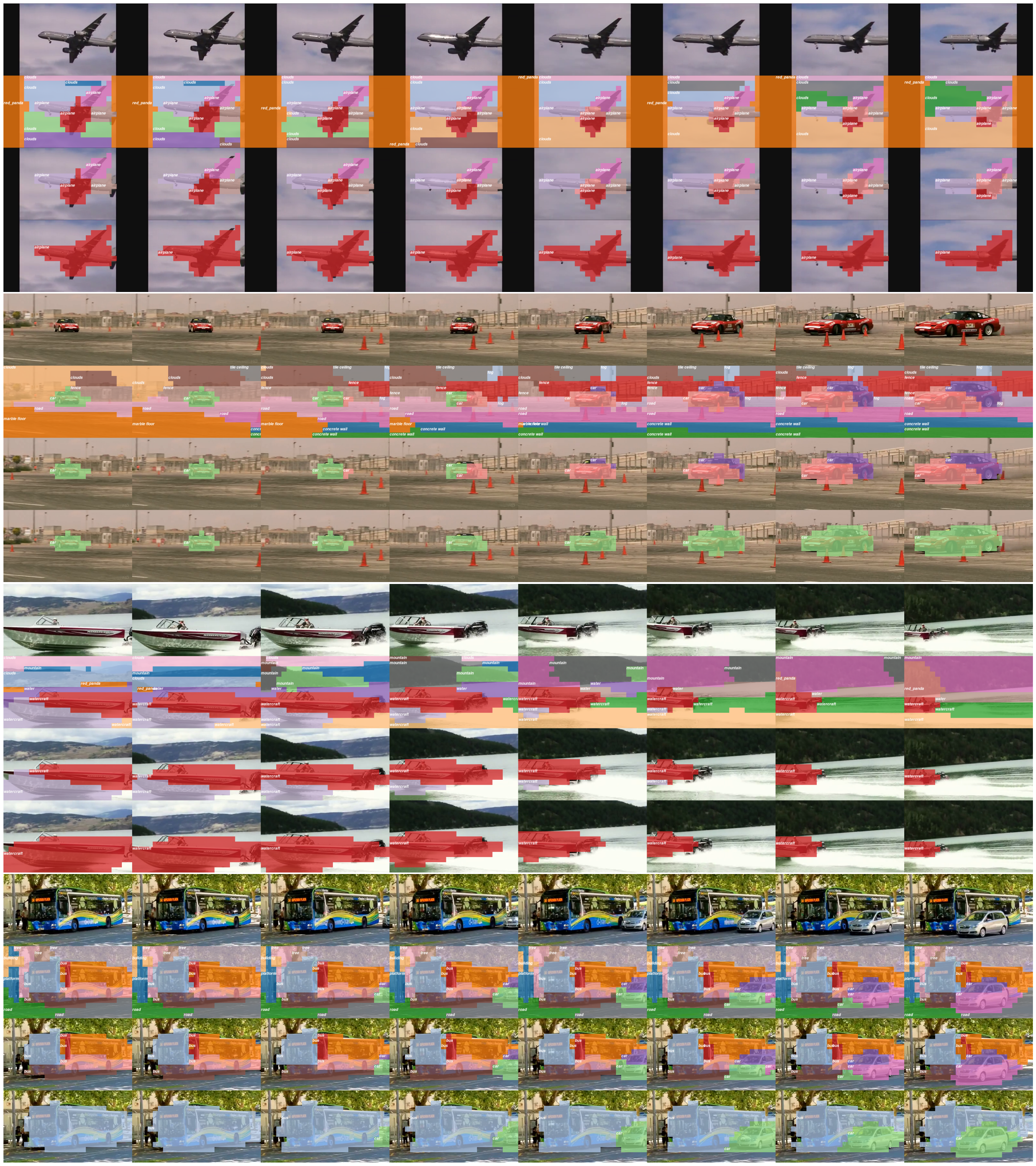
Figure 5. More qualitative examples on ImageNet-VID dataset. The four rows in each example correspond to the input frames, labeled localization, objects of interests and merged objects.

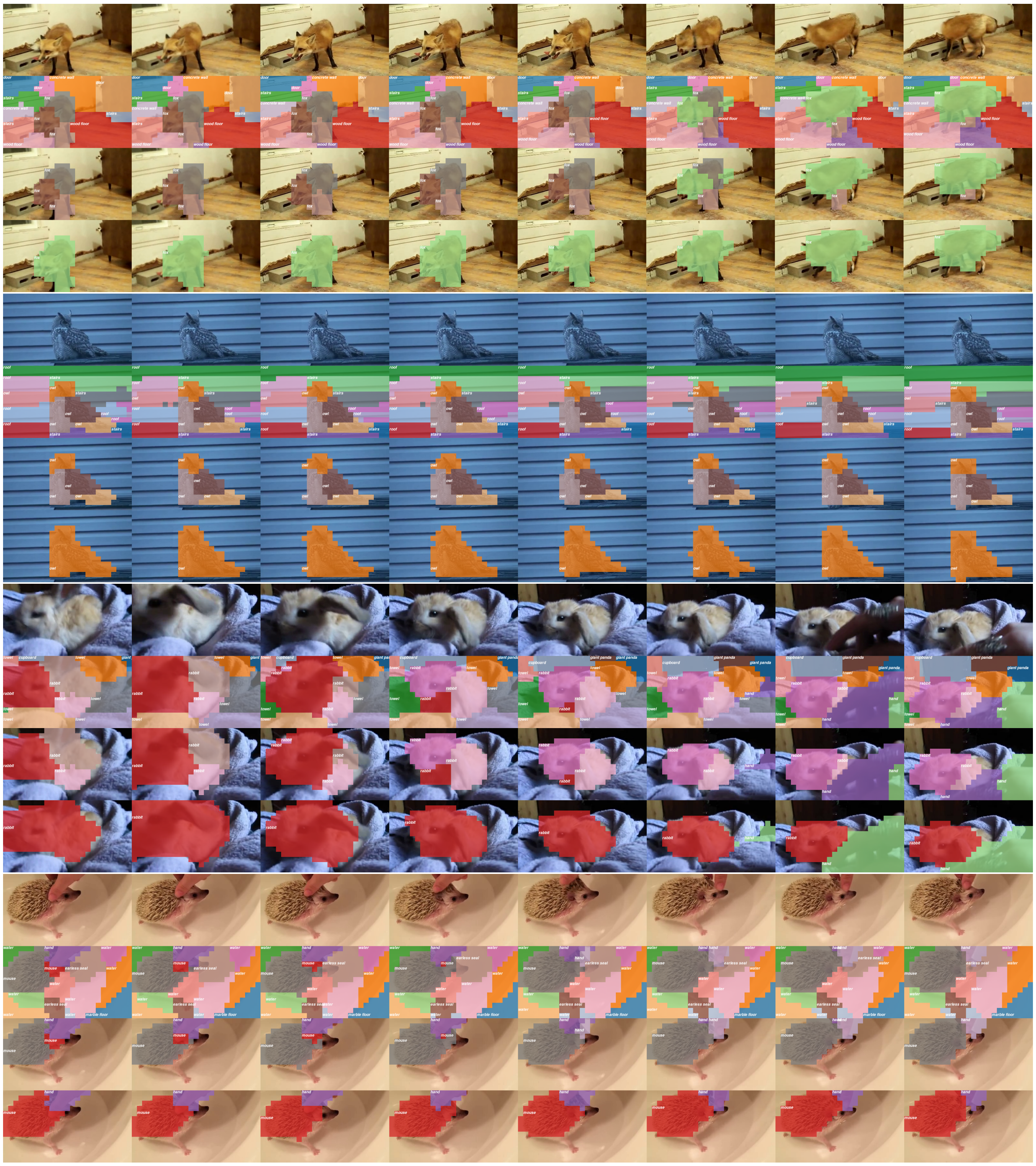

Figure 6. More qualitative examples on YouTube-VIS dataset. The four rows in each example correspond to the input frames, labeled localization, objects of interests and merged objects.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-scale Hierarchical Image Database. In *CVPR*, 2009.

[3] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *ICCV*, 2017.

[4] Geoffrey Hinton. How to Represent Part-whole Hierarchies in a Neural Network. *Neural Computation*, pages 1–40, 2022.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[8] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019.