Improving Closed and Open-Vocabulary Attribute Prediction using Transformers - Supplementary Materials

Khoi Pham[1], Kushal Kafle[2], Zhe Lin[2], Zhihong Ding[2], Scott Cohen[2], Quan Tran[2], and Abhinav Shrivastava[1]

[1] University of Maryland, College Park
{khoi,abhinav}@cs.umd.edu
[2] Adobe Research
{kkafle,zlin,zhding,scohen,qtran}@adobe.com

Contents

1 VAW experiments

1.1 RN50-Context Baseline

In our main paper, we introduce a new baseline for the VAW benchmark which we name RN50-Context. Basically, RN50-Context has the same architecture as

the RN50-Baseline used in [11] that first fuses the image features with the object word embeddings features before classification. However, instead of predicting attributes for every object independently, RN50-Context (1) takes in the whole input image to produce the whole image feature maps, (2) uses RoIAlign to pool features for every object residing in the image, then (3) predicts attributes for every object at once. Figure 1 demonstrates how RN50-Context performs attribute prediction. For illustration purpose, the figure only shows how this is done for one object. The same set of computations can be parallelized and applied for all other objects in a given image, which makes this model more efficient than the RN50-Baseline in [11].

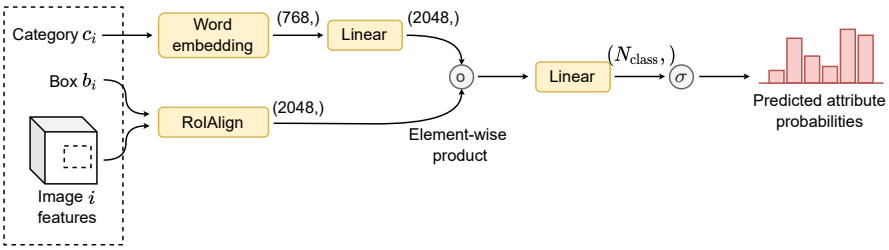

Fig. 1. Overview diagram of the RN50-Context baseline used in VAW.

1.2 TAP's detailed performance breakdown

We present in Table 1 the detailed breakdown of our model performance on each attribute type in the VAW dataset. It can be seen that our model is better than the SOTA model SCoNE [11] in almost all attribute types, with the largest improvement lies in *action* categories. This shows that our model utilizes context information effectively because *action* usually requires more context than the others. Regarding basic attribute types such as *color, material, shape, size*, our improvement is more modest, especially in *shape* performance where we achieve only comparable result with SCoNE. In SCoNE [11], the authors also use low-level image ResNet features (from early ResNet blocks) and explain that this improves their mAP on low-level attributes including *color, material, shape, texture*. Therefore, we will explore using ResNet features from its early blocks in our future work to get even better results.

1.3 Qualitative results

We show several examples of how our model predicts attributes on images in the VAW dataset in Figure 2. We number the images from left to right, then from top to bottom. In image 1, the model predicts perfectly attributes for *sky* and *boat*, while for *person* it incorrectly predicts *old* and *talking*. It is possible

Table 1. Detailed breakdown of performance on VAW.

Methods	Overall				Class imbalance (mAP)			Attribute types (mAP)							
	mAP	mR@15	mA	F1@15	Head	Medium	Tail	Color	Material	Shape	Size	Texture	Action	State	Others
RN50-Baseline [11]	63.0	52.1	68.6	63.9	71.0	59.6	43.2	57.4	66.0	64.7	65.2	61.3	53.6	61.2	66.2
SCoNE [11]	68.4	58.4	71.5	70.2	76.4	65.0	48.3	69.3	75.3	**68.0**	70.2	66.8	61.2	64.8	69.0
RN50-Context	67.3	54.1	69.3	66.1	71.3	67.3	52.1	61.8	71.1	60.9	63.1	64.8	67.5	63.7	69.8
TAP [**Ours**]	**73.4**	**63.3**	**73.5**	**71.1**	**77.6**	**72.9**	**58.8**	**71.0**	**77.3**	67.9	**71.6**	**70.4**	**73.4**	**69.7**	**74.5**

that the model sees a group of people sitting close together so it predicts *talking*. In image 2, the model incorrectly predicts *lying* for *dog* with lower confidence, but predicts all correct attributes for *grass*. In image 3, *cloudy* is incorrectly predicted for *sky*, however, due to the brightness of the image it is difficult to see if the sky is truly cloudy. For image 4, the *wall* is incorrectly predicted with color *brown*, however, we believe the color of the *wall* here still bears a small resemblance to *brown*. In the last image, the model incorrectly predicts *grazing* for *cows*. Because *grazing* is very oftenly labeled for *cows* in the dataset, we believe the model has learned a small bias towards outputting this action for *cows*.

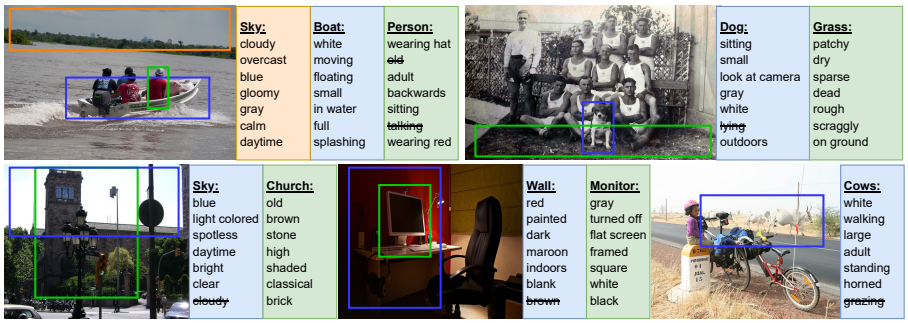

Fig. 2. Examples of our model predictions on the VAW dataset.

2 LSA open-vocabulary experiments

2.1 Prompts

In the main paper, we explain how we use an ensemble of multiple prompts to generate text embeddings of the attribute classes for our OpenTAP model and for the CLIP baselines. As suggested by the authors from the CLIP paper [13] and also from our empirical results, we construct the ensemble of these prompts in the embedding space rather than the probability space. Here, we will list all prompts that we use.

OpenTAP and 'CLIP (attribute prompt)' baseline Our OpenTAP model and the 'CLIP (attribute prompt)' baseline (Sect. 5.2) both use the following same set of prompts:

- `<attr>.`
- `A photo of <attr>.`
- `A photo of something <attr>.`
- `A photo of something that is <attr>.`
- `A cropped photo of <attr>.`
- `A cropped photo of something <attr>.`
- `A cropped photo of something that is <attr>.`

'CLIP (object-attribute prompt)' baseline The 'CLIP (object-attribute prompt)' baseline (Sect. 5.2) uses the following set of prompts that also mention the object category name. For location and interaction classes (*e.g.*, *on table, carrying bag*), we only use the prompts 2, 4, and 6 because the other prompts produce non-sensical sentences (*e.g.*, we want *person carrying bag* instead of *carrying bag person*).

1. `<attr> <obj>.`
2. `<obj> <attr>.`
3. `A photo of <attr> <obj>.`
4. `A photo of <obj> <attr>.`
5. `A cropped photo of <attr> <obj>.`
6. `A cropped photo of <obj> <attr>.`

For the 'CLIP (combined prompt)' baseline, we take the weighted average of the output of the 'CLIP (attribute prompt)' and the 'CLIP (object-attribute prompt)'. The weights used for averaging are chosen based on the validation set.

2.2 Errors of CLIP baselines

In the main paper, we explain that using only the object attribute prompts (numbered 1-6 above) returns drastically low accuracy due to CLIP being highly attentive to the object, almost ignores the attribute, and unable to detect non-sensical object-attribute pairs, *e.g.*, for an image of a *hydrant* with the text *A photo of a happy hydrant*, CLIP still returns a high similarity score even though *happy hydrant* is non-sensical, sometimes ranking even higher than true and valid combinations, such as *happy person*. In other words, just the existence of the object in the image alone already leads to a high similarity score. This characteristic of CLIP messes up the returned ranklists of this baseline. To alleviate this, we combine the predicted score of the 'CLIP (object-attribute prompt)' and 'CLIP (attribute prompt)' baseline, as we find this suppresses the object part and allows CLIP to focus more on the attribute aspects (*i.e.*, adjective and verb).

In Figure 3, we show failure examples of the returned ranklist of the 'CLIP (object-attribute prompt)' baseline for attributes *jumping, happy,* and *using laptop*. In almost all cases, the top retrieved images simply contain only the object of interest and almost totally ignore the attribute.

In Figure 4, we present the top retrieval results for the same set of attributes but using the 'CLIP (combined prompt)' baseline. This shows that we can alleviate the aforementioned problem by combining with the 'CLIP (attribute prompt)'. Throughout the figure, it can be seen that the retrieved results are much more accurate. However, the 'non-sensical object-attribute pairs' problem still occurs. For *jumping*, combinations such as, *jumping yard* and *jumping snow* are still at the top of the ranklist even though these pairs are non-sensical. This might be due to the 'combined model' detecting the attribute and objects cues separately (in the *yard* and *snow* image, the visual cues for *jumping* belongs to the animal and not the *yard* and *snow*). The same thing happens for *happy shirt* on the 2nd row where we know that *happy shirt* does not make sense, but the model still returns it because it finds visual cues for *happy* from the *girl*. For *using laptop*, the same thing happens for the non-sensical pair *table - using laptop*.

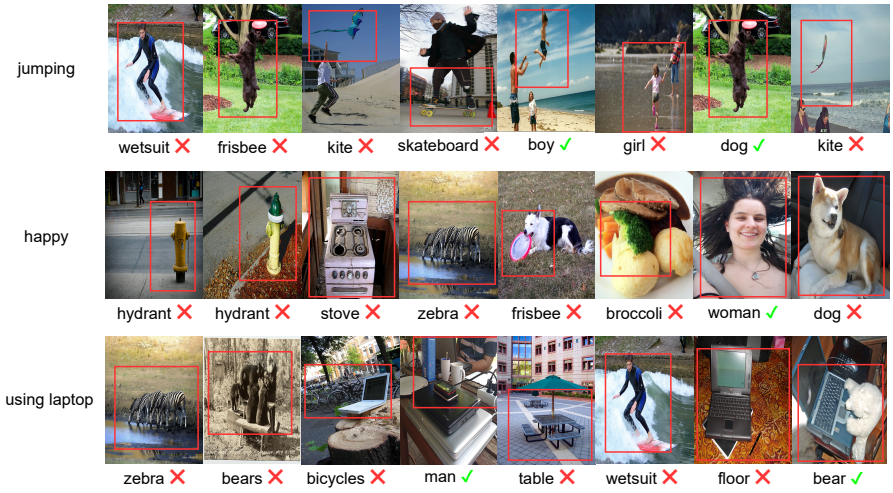

Fig. 3. Top image retrieval results of the 'CLIP (object-attribute prompt)' baseline for 3 attributes *jumping, happy,* and *using laptop*, with incorrect predictions marked with ✗ and correct predictions marked with ✓. Below each image is the category of the object whose attributes are being predicted for.

2.3 OpenTAP qualitative results

In Figure 5, we show the top retrieval results for the same set of attributes as above, including *jumping, happy,* and *using laptop*. Even though OpenTAP uses the same text embeddings as the 'CLIP (attribute prompt)' baseline, OpenTAP is still able to rank the correct images on top. This shows that CLIP text

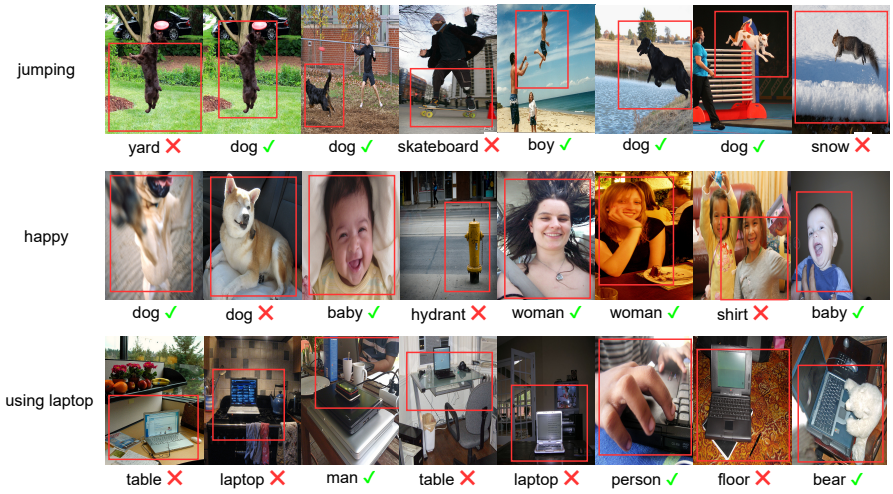

Fig. 4. Top image retrieval results of the 'CLIP (combined prompt)' baseline (the best overall CLIP baseline) for 3 attributes *jumping, happy,* and *using laptop*, with incorrect predictions marked with ✗ and correct predictions marked with ✓. Below each image is the category of the object whose attributes are being predicted for.

embeddings do really contain knowledge about attributes, however, using them out of the box is challenging due to the aforementioned problems. By training OpenTAP on LSA with our training scheme, OpenTAP is able to use CLIP text embeddings for predicting attributes much more accurately.

2.4 Baseline choices

Attribute prediction for objects in the wild is very underexplored, especially in the open-vocabulary setting. Open-vocabulary recognition has been studied before, but mostly for object recognition and detection and not for attribute prediction. In order to select good baselines for our LSA experiment, we look for SOTA vision-language models since these models have been trained on a large corpus of image-text pairs that already include attribute concepts. CLIP is the best candidate as a baseline since (1) CLIP has been trained on enormous amount of data (400M image-text pairs) which makes it fair to expect that CLIP should perform well on attribute recognition, and (2) CLIP is easy and fast to use in a retrieval/classification setting where we want to output prediction scores for as many as 9000 attributes in LSA (CLIP uses one single dot product to output all scores). In addtion, we believe the CLIP baselines that are presented in the main paper could be further improved by fine-tuning CLIP image encoder on LSA.

Other baselines that we can use are SOTA vision-language pre-trained or VQA models [9,7,15] . However, these models are extremely expensive to evaluate

on LSA. For example, given an image, for every attribute, we have to construct a different text query/question then perform model inference from the beginning.

In addition to LSA, it is also possible to evaluate OpenTAP on other benchmarks to better showcase its ability and compare against other methods in understanding attributes. We have demonstrated OpenTAP ability on another benchmark that is the HICO dataset [3]. Another interesting benchmark is the recently proposed GPV-Web10k dataset [8] that contains images with 298 verbs and 148 adjectives annotations. We leave this as future work.

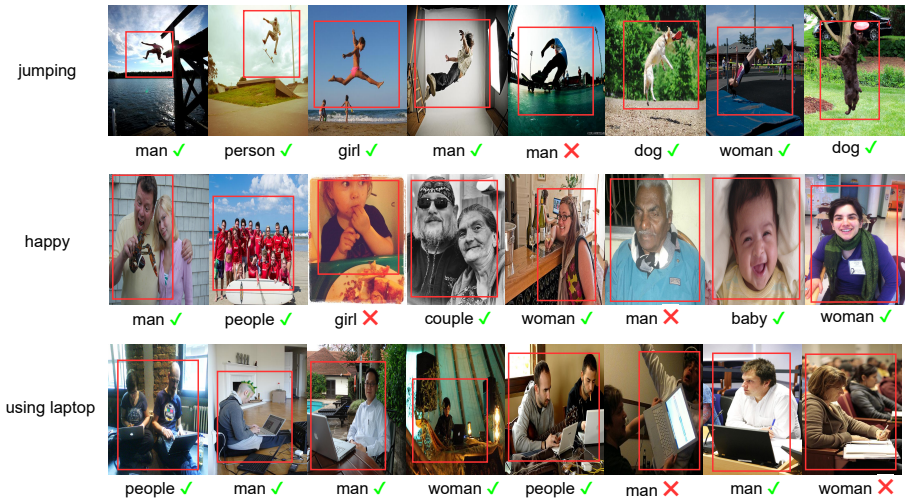

Fig. 5. Top image retrieval results of our OpenTAP model for 3 attributes *jumping, happy,* and *using laptop*, with incorrects marked with ✗ and corrects marked with ✓. Below each image is the category of the object whose attributes are being predicted for.

3 LSA dataset

3.1 Attribute extraction example

We provide in Fig. 6 two examples of how we extract attributes from captions, with one case having grounding information for the objects and one without.

3.2 Localized Narratives data processing

Mouse trace to bounding box: in Localized Narratives [12], every word (or utterance) in a caption is accompanied with a mouse trace segment that describes where the annotator moves their mouse cursor while speaking the

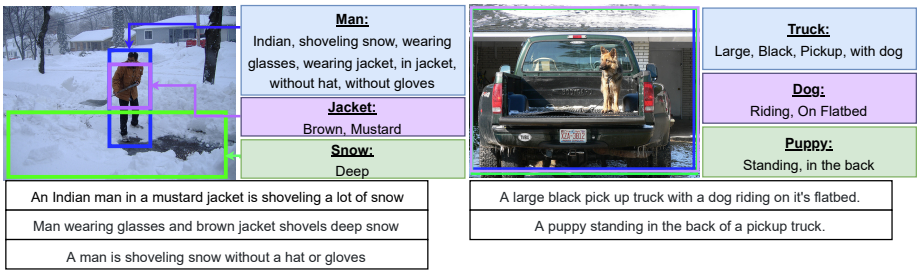

Fig. 6. Extracted attributes example. Objects and attributes parsed using grounded (left, from Flickr30k) and ungrounded captions (right, from COCO).

word. Therefore, this mouse trace segment describes roughly the image location of any words. In our work, after we apply our parsing algorithm to extract objects in a given caption, we follow Sec 4.3 in [2] to convert the mouse trace of the parsed objects to their bounding boxes. We also apply temporal padding as in [2] to handle cases when an annotator speaks before (or after) their mouse is actually moved. In addition, we also enlarge the converted bounding box by 40% vertically and horizontally as we notice a large number of cases where the annotator only circles their mouse cursor around a small region of the object.

In Figure 7, we show some examples of boxes converted from mouse trace in the Localized Narratives dataset. It can be seen that these boxes are not accurate as they do not fit exactly to the object. Some boxes occupy a large region around the object, while some boxes only occupy the object partially. Despite this, these boxes still provide important object localization information.

Fig. 7. Localized Narratives's converted bounding boxes from mouse trace.

3.3 Attribute statistics

We provide in Table 2 the number of attributes in the adjective, verb, interaction, and location subcategories in our seen set $\mathcal{C}_s$ and unseen set $\mathcal{C}_u$ of the open-vocabulary experiment. Because of the compositional nature of the location classes, the number of attributes in this subcategory could be extremely large (*e.g.*, for 4 basic prepositions *on, in, under, next to* and N different object categories, the total number of location classes could go up to $N \times 4$). Therefore, for

location classes in our seen set, we only keep those that involve common object categories (*i.e.*, the top-1000 most frequent object categories across LSA).

Table 2. Number of attributes in each type.

Attribute types	**# of classes in $\mathcal{C}_s$**	**# of classes in $\mathcal{C}_u$**
Adjective	1251	1058
Verb	950	757
Interaction	1278	906
Location	2047	1291
Total	5526	4012

From Figure 8 to Figure 11, we also provide the long-tail distribution of attributes in $\mathcal{C}_s$ and $\mathcal{C}_u$, belong to each type (adjective, verb, interaction, location), across the whole LSA dataset.

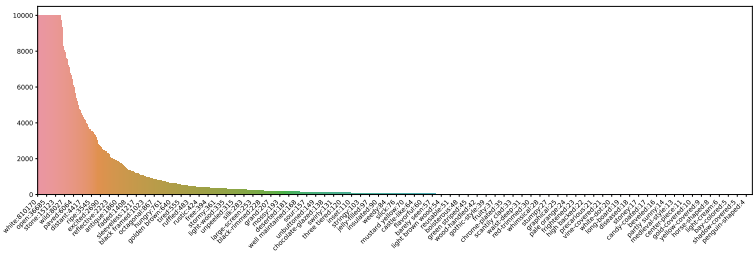

Fig. 8. Distribution of adjective attributes.

4 Implementation details

4.1 TAP architecture: multi-modal Transformer

We use 512 as the input dimension to the multi-modal Transformer. At input, to map the visual tokens $\{v_i\}$ and query tokens $\{h_j\}$ to the same embedding space that has dimensionality of 512, we respectively use two FC layers FC_{img} and FC_{query}. FC_{img} maps from 2048-d ResNet features, and FC_{query} maps from 768-d BERT word embeddings. Inside the multi-modal Transformer, we use the same inner-layer dimension 2048 for the feedforward network (FFN) as in [14]. Similar to [1], we use additive dropout 0.1 after every multi-head attention and FFN before layer normalization. All of the Transformer weights are initialized using Xavier init [5].

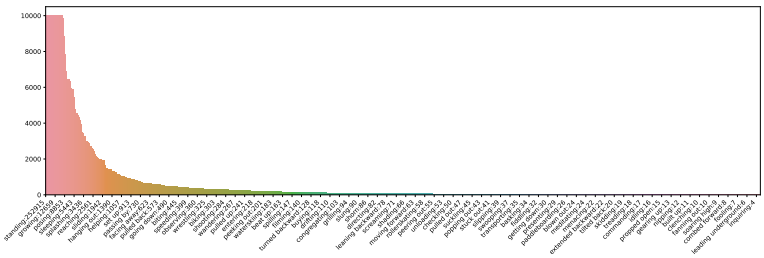

Fig. 9. Distribution of verb attributes.

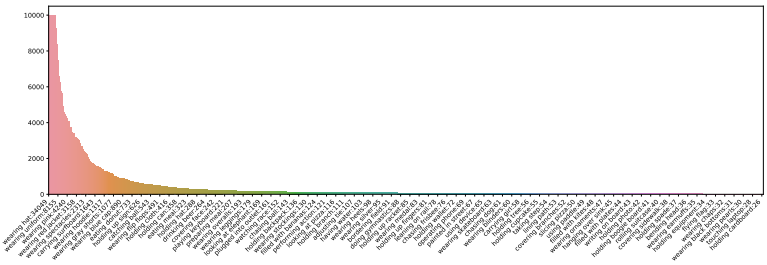

Fig. 10. Distribution of interaction attributes.

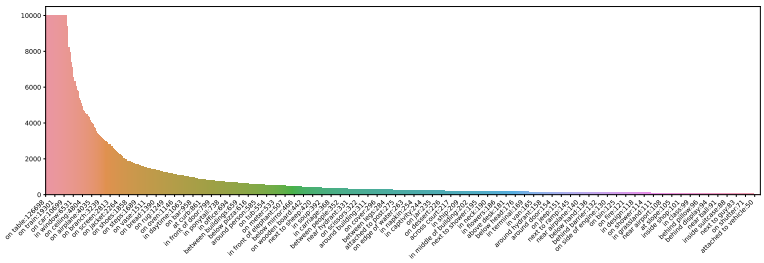

Fig. 11. Distribution of location attributes.

4.2 TAP object grounding

We use the object grounding loss to train the model to softly localize the image region for any object of interest. This allows the model to know where to attend to for object without bounding box (*e.g.*, objects from unaligned image-text pairs in MS-COCO Captions [4]), making the model able to learn attribute prediction even from ungrounded objects. We leverage the portion of our data with known grounding information to train with this loss. We randomly remove the box information of an object at input with probability 15%, *i.e.* we set the box coordinates of an object to be the top-left and bottom-right corner of the entire image to remove its location information. We avoid instances where this might cause ambiguity, *i.e.* when multiple objects of the same type are present in the image which makes the model unable to know which instance is being referred to. In addition, for a training image with N objects, we also randomly sample a subset of $k \leq N$ objects to be used for training in each iteration. This is to prevent the model from learning spurious correlations based on occurrence of objects in the context.

4.3 Training

LSA training we train our model using AdamW [10] with learning rate $1e^{-4}$. For the ResNet-50 backbone, it is trained using learning rate $1e^{-5}$ with frozen BatchNorm layers. We keep our query sequence at maximum length of 40. Training image is resized to 512×512 which results in the visual sequence length $L_v = 16 \times 16 = 256$. For image augmentation, we apply random flipping and cropping such that none of the sampled boxes are cropped. In the object grounding loss, we use $\tau = 0.02$. In the BCE classification loss (formula (7) in the main paper), we treat missing labels as negatives with a small weight of 0.02, while computing p_c and n_c in the same way as [11]. The model is trained for 34 epochs with learning rate reduced by a factor of 10 at epoch 28 and 32. We use batch size 96. The whole training takes approximately 30 hours using 4 NVIDIA GeForce GTX 1080 Ti GPUs.

VAW finetuning All attribute labels in VAW are present in the seen attribute set $\mathcal{C}_s$ of LSA, hence we can use our model with the closed-set classifier head (TAP) to fine-tune and test on the VAW benchmark. TAP is fine-tuned with learning rate $1e^{-5}$, while its ResNet backbone is fine-tuned with learning rate $1e^{-6}$. We fine-tune for 16 epochs with learning rate multiplied by 0.3 at epoch 10 and 14.

HICO finetuning Because the majority of labels in HICO are not present in the seen attribute set $\mathcal{C}_s$ of LSA, we use our model with the open-set classifier head (OpenTAP) for this experiment. To initialize classifier for all human-object interaction (HOI) classes, we use the prompt 'a person `<interaction>` `<object>`' (*e.g.*, 'a person riding horse') and generate CLIP text embeddings of

them. This initialization step is the same with [6]. We then fine-tune our model with learning rate $1e^{-5}$, the ResNet backbone with learning rate $1e^{-6}$, and the HOI classifiers with learning rate $1e^{-4}$. We fine-tune for 18 epochs with learning rate reduced by 10 at epoch 14.

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020) 9
2. Changpinyo, S., Pont-Tuset, J., Ferrari, V., Soricut, R.: Telling the what while pointing to the where: Multimodal queries for image retrieval. arXiv preprint arXiv:2102.04980 (2021) 8
3. Chao, Y.W., Wang, Z., He, Y., Wang, J., Deng, J.: Hico: A benchmark for recognizing human-object interactions in images. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1017–1025 (2015) 7
4. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015) 11
5. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256. JMLR Workshop and Conference Proceedings (2010) 9
6. Jin, Y., Chen, Y., Wang, L., Wang, J., Yu, P., Liang, L., Hwang, J.N., Liu, Z.: Decoupling object detection from human-object interaction recognition. arXiv preprint arXiv:2112.06392 (2021) 12
7. Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I., Carion, N.: Mdetr-modulated detection for end-to-end multi-modal understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1780–1790 (2021) 6
8. Kamath, A., Clark, C., Gupta, T., Kolve, E., Hoiem, D., Kembhavi, A.: Webly supervised concept expansion for general purpose vision models. arXiv preprint arXiv:2202.02317 (2022) 7
9. Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., et al.: Oscar: Object-semantics aligned pre-training for vision-language tasks. In: European Conference on Computer Vision. pp. 121–137. Springer (2020) 6
10. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 11
11. Pham, K., Kafle, K., Lin, Z., Ding, Z., Cohen, S., Tran, Q., Shrivastava, A.: Learning to predict visual attributes in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13018–13028 (2021) 2, 3, 11
12. Pont-Tuset, J., Uijlings, J., Changpinyo, S., Soricut, R., Ferrari, V.: Connecting vision and language with localized narratives. In: European Conference on Computer Vision. pp. 647–664. Springer (2020) 7
13. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020 (2021) 3
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017) 9
15. Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., Choi, Y., Gao, J.: Vinvl: Making visual representations matter in vision-language models. arXiv preprint arXiv:2101.00529 (2021) 6