Zero-shot Object Detection Through Vision-Language Embedding Alignment

Anonymous

Paper ID: S18201

***Abstract*—Recent approaches have shown that training deep neural networks directly on large-scale image-text pair collections enables zero-shot transfer on various recognition tasks. One central issue is how this can be generalized to object detection, which involves the non-semantic task of localization as well as semantic task of classification. To solve this problem, we introduce a vision-language embedding alignment method that transfers the generalization capabilities of a pretrained model such as CLIP to an object detector like YOLOv5. We formulate a loss function that allows us to align the image and text embeddings from the pretrained model CLIP with the modified semantic prediction head from the detector. With this method, we are able to train an object detector that achieves state-of-the-art performance on the COCO, ILSVRC, and Visual Genome zero-shot detection benchmarks. During inference, our model can be adapted to detect any number of object classes without additional training. We also find that standard object detection scaling can transfer well to our method and find consistent improvements across various scales of YOLOv5 models and the YOLOv3 model. Lastly, we develop a self-labeling method that provides a significant score improvement without needing extra images nor labels.**

***Index Terms*—Zero-shot Learning; Object Detection; Vision-Language Alignment**

I. Introduction

Zero-shot detection (ZSD) [1] aims to train a model to detect unseen objects. Despite its difficulty, zero-shot object detection has many applications. For example, to develop a Chimpanzee detector [2], animal scientists needed to collect specialized training data and exhaustively label for Chimpanzees, but with a generally trained zero-shot object detector they could simply detect using new reference embeddings.

Typical object detectors are trained on data with a fixed number of categories. The PASCAL VOC dataset [3] has 20 classes, while the COCO dataset [4] has 80 classes. These datasets have inspired many important object detection methods such as DPM [5], R-CNN [6]–[8], SSD [9], and YOLO [10]–[12].

While research in detecting rare categories with higher accuracy has shown extensive progression, models continue to score considerably lower for rare objects [13], [14]. Furthermore, gathering detection data for a large number of categories is difficult due to the long-tailed nature of objects and because exhaustively labeling many class types is expensive. In contrast, gathering text image pairs is a far easier task as these images can be sourced from the internet to assemble massive training datasets such as YFCC100M [15]. Using this dataset, Radford *et al.* [16] demonstrated impressive zero-shot capabilities across a wide array of computer vision datasets with varying category granularity such as [17]–[20], but they have not explored the zero-shot capabilities for object detection. In this paper we devise a method to transfer this zero-shot capability to object detection.

To this end, we propose a method for adapting a one-stage detector to perform the ZSD task through aligning detector semantic outputs to embeddings from a trained vision-language model. Compared to previous ZSD works that only learn from text embedding alignment, such as [1], [21]–[27], we also consider image embedding alignment in our proposed method. We find that this addition of image embeddings to our proposed loss function provides a significant improvement to model performance. To obtain image embeddings, we crop ground truth bounding boxes and feed them through the CLIP [16] image encoder. Model class outputs are then aligned using an $\mathcal{L}_1$ loss function. Furthermore, we create a new post-processing operation tailored to the ZSD task for YOLO [10] style models to better detect unseen classes in scenes with both seen and unseen classes. Lastly, we also show that our method achieves consistent improvements on the different YOLOv5 network sizes. In summary, our contributions are:

- We show it is possible to generalize the capability of the pre-trained vision-language model CLIP [16] to address zero-shot detection. Specifically, we train ZSD-YOLO, our one stage zero-shot detection model [12] that aligns detector semantic outputs to embeddings from a contrastively trained vision-language model CLIP [16].
- We develop a self-labeling data augmentation method that provides 0.8 mAP and 1.7 mAP improvement on the 48/17 [1] and 65/15 [28] COCO ZSD splits respectively without needing extra images nor labels.
- We propose a ZSD tailored post-processing function that provides a 1.6 mAP and 3.6 mAP improvement on the 48/17 [1] and 65/15 [28] COCO ZSD splits respectively.
- We demonstrate that there are consistent improvements for the proposed ZSD-YOLO model with different network sizes which benefits model deployment under varying computational constraints.

II. Related works

Zero-shot learning. Zero-shot learning (ZSL) [29] aims at developing a model that can identify data class types whose categories are not seen during training. In ZSL [30], there are still some labeled training instances in the feature space and

the categories of these instances are referred to as seen classes and the unlabeled instances are considered unseen classes. Prior works [31]–[34] in ZSL focus on attribute learning which focuses on learning to recognize the properties of objects. However, these approaches treat all dimensions of the attribute space equally, which is sub-optimal because certain attributes are more important than others for a given task. More recently, CLIP [16] uses contrastive learning that jointly trains an image encoder and a text encoder on a subset of [15] to create a network which predicts the correct pairings of batched image-text training examples. During inference, the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's unseen classes.

Zero-shot detection. Zero-shot detection (ZSD) was introduced through Bansal *et al.* [1] and describes the task of detecting objects that have no labeled samples in the training set. Bansal *et al.* [1] established a baseline model through aligning model outputs to word-vector embeddings through linear projection and created the 48/17 benchmarking split based on the COCO [35] detection dataset. Zhu *et al.* [36] used the YOLOv1 [10] detector to improve zero-shot object recall. Li *et al.* [22] developed an attention mechanism to address zero-shot detection. Rahman [28] adopts the RetinaNet detector for zero-shot detection, using a polarity loss to increase the distance between class embeddings and introduced the 65/15 ZSD split on the COCO dataset. Zhao *et al.* [24] used a GAN to synthesize the semantic representation for unseen objects to help detect unseen objects. Zhu *et al.* [37] proposed DELO that synthesizes the visual features for unseen objects from semantic information and incorporates the synthesized features for the detection of seen and unseen classes Zheng *et al.* [27] proposed a method to address the task of zero-shot instance segmentation using a method that consists of zero-shot detector, semantic mask head, background aware RPN, and synchronized background strategy. Hayat *et al.* [38] requires knowledge of which classes will be used for unseen detection while training as well as their semantic embeddings, while ours and prior works such as [1], [21], [27], [28] can train without having to set unseen classes in training and can therefore run inference on a new set of unseen objects without any retraining. For this reason our paper as well as previous zero-shot detection papers such as [27] have not compared with their scores in benchmarking.

Open Vocabulary Detection. Very recently, Zareian *et al.* [39] created a new dataset split of the COCO dataset based on the 2017 train and validation splits and introduced a task known as open-vocabulary detection (OVD). This new task does not eliminate instances of unseen classes in the detector training data. Gu *et al.* [40] has explored the idea of improving open vocabulary detection, based on knowledge distillation and the two-stage Mask-RCNN [41] model. Given that the task of OVD is relatively similar to the task of ZSD, we explain how previously published ZSD papers have adhered to certain protocols and how the new OVD task differs from zero-shot detection in the Section C of the appendix.

In contrast to the prior works, our paper explores aligning one stage detector semantic outputs to CLIP embeddings to address the task of ZSD. To this end, we develop a new training algorithm for embedding alignment and modify the typical YOLOv5 model architecture to support ZSD. To optimize our method, we tailor the typical YOLOv5 post-processing method to ZSD and develop a self-labeling data augmentation method to expand the knowledge of a zero-shot detector without additional data. We also explore model scaling, another crucial aspect of traditional detection systems that allows a model architecture to fit various applications with different computational constraints.

III. Method

The main process for embedding alignment involves altering the YOLOv5 [12] detection head to produce a semantic embedding that mimics the CLIP model's outputs for every detection anchor. To approximate the embedding space of CLIP with our model we minimize alignment losses for both text and image embeddings that align detector semantic outputs at each given anchor with the text and image encoder outputs of CLIP [16]. As shown in Figure 1, we propose aligning vision and language embeddings from a model such as CLIP to adapt an one-stage detector such as YOLOv5 [12] for ZSD.

Problem definition. The problem of ZSD as defined by [1] involves detecting unseen classes $\mathcal{U}$ using training images D_{train} containing only labeled seen classes $\mathcal{S}$ and unlabeled instances of other objects in the background denoted as $\mathcal{O}$. Note that $\mathcal{U}$, $\mathcal{S}$, $\mathcal{O}$ are pairwise disjoint. To detect novel classes, we align detector semantic outputs with the embedding space of a generalized model, in our case CLIP, by approximating its visual-semantic vector embedding space. As shown in [27], zero-shot prediction without some related prior knowledge is impossible, and in our case, the prior knowledge is provided by the visual-semantic embedding space of CLIP optimized for classes $\mathcal{C} = \mathcal{S} \cup \mathcal{U} \cup \mathcal{O}$.[1] During inference, our model produces semantic outputs $\mathcal{M}$ for each detection anchor that approximate the visual-semantic embedding space of CLIP by mimicking its encoders. These model semantic outputs are compared with the CLIP generated text embeddings of class names of $\mathcal{U}$ to detect instances of $\mathcal{U}$ for ZSD on the set D_{test}.

Choice of base model. We choose the YOLOv5 [12] model family as our base model for three main reasons. Firstly, since typical detection models are often modified to produce only a single bounding box regression output per anchor for the task of ZSD, we believe YOLOv5 [12] is more suited for the task of ZSD as it already uses only a single bounding box regression output per anchor whereas a majority of anchor based object detectors use one bounding box regression output per class. Second, YOLOv5 [12] models and other YOLO type models generally provide the best inference speed, mAP trade-off in standard supervised object detection [42] and we find

[1]To our knowledge all previous encoders used by ZSD methods were optimized for a similarly large, general class space

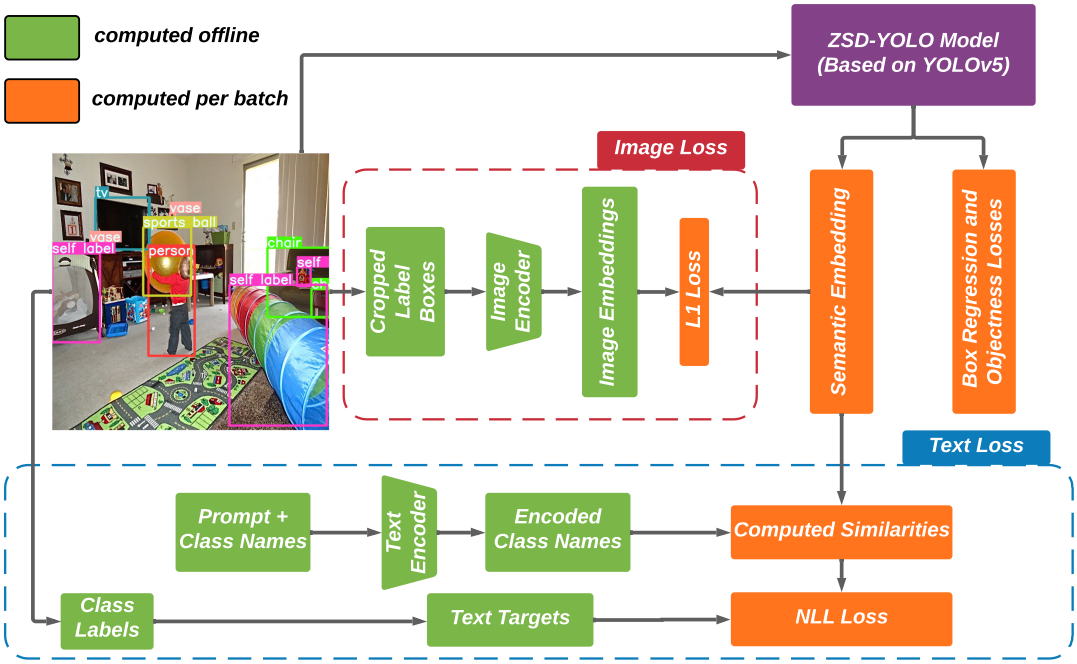

Fig. 1. **An overview of our proposed training method of ZSD-YOLO.** Our method aligns detector semantic outputs to vision and language embeddings from a pretrained Vision-Language model such as CLIP. We modify YOLOv5 to replace typical class outputs with a semantic output with shape equal to the CLIP model embedding size and then align predicted semantic outputs of positive matched anchors with corresponding ground truth text embddings with a modified crossentropy loss described in section III-A. Image Embeddings of positively matched anchors are aligned using a modified $\mathcal{L}_1$ loss function described in section III-B. Best viewed in color.

these advantages transfer well to the zero-shot detection task. Third, unlike a majority of two-stage detection models, one-stage models like YOLOv5 [12] compute class confidences for every anchor irrespective of objectness scores, a function we leverage to create a post-processing function that signficantly improves ZSD performance.

Pretraining procedure. To produce initial weights for ZSD training we train a standard YOLOv5 model of the same scale on the same seen class split with class specific annotations so that the model backbone can be initialized with class specific information when initialized for ZSD. All of these pretraining procedures are initialized with random weights and trained only on their corresponding seen label dataset. The main purpose of this process is to accelerate the training convergence rate of a zero-shot detection model.

A. Aligning Text Embeddings

The process of aligning text embeddings involves the alignment of detector semantic outputs with the target text embeddings generated by CLIP [16]. An overview of this process is highlighted in the dotted blue box found in Figure 1. To compute the loss for text embeddings, we first generate the seen class text embeddings $\mathcal{T}$ by feeding every seen class name into the prompt "A photo of $\{$class$\}$ in the scene". The next step in computing the text loss is to extract model semantic class outputs of positive anchors $\mathcal{M}_g$ represented as "Semantic Embedding" in Figure 1, selected by the standard YOLOv3 [43] procedure based on anchor box Intersection over Union (IoU) with ground-truth labels. We then generate a similarity matrix using a cosine similarity computation relating model output embeddings $\mathcal{M}_g$ and seen text embeddings $\mathcal{T}$ followed by a softmax with temperature τ. We exponentiate and set τ as a learnable parameter in order to fit an optimal temperature similar to the original CLIP [16] training, though in our final model training τ is fixed. The full computation to generate similarity matrix $\mathbf{z}$, representing the "Computed Similarities" in Figure 1 can be written as:

$$\mathbf{z} = softmax\Big(\frac{\mathcal{M}_g\mathcal{T}^\top}{\|\mathcal{M}_g\|\|\mathcal{T}\|}e^\tau\Big). \tag{1}$$

To compute the the full text loss $\mathcal{L}_{\text{text}}$, we gather ground truth box label classes corresponding to each positively matched anchor to create one-hot encoded labels, y with shape identical to the generated similarity matrix $\mathbf{z}$. The final text loss is computed with a negative log-likelihood (NLL) function measuring distance between $\mathbf{z}$ and y.

B. Aligning Image Embeddings

The process of aligning image embeddings involves aligning model semantic outputs of each positive anchor, the same as those selected for text embedding alignment, with the corresponding image embedding. An overview of the image loss computation is outlined in the red dotted box in Figure 1. We first crop then preprocess each ground truth bounding box and apply the CLIP [16] image encoder to this region offline prior to training. The preprocessing applied to the

images is adapted from the official CLIP preprocessing with the typical center crop replaced with a resize. During training, all semantic outputs of anchors matched to a ground truth bounding box, the same as those selected for text-embedding alignment, are aligned with their corresponding ground-truth image embedding generated by the CLIP model. Our final image loss function is calculated using a $\mathcal{L}_1$ loss and while we experimented with a smooth exponentiation factor for the distance function in our hyperparameter search, these did not produce any statistically significant improvements. In Figure 1, let $\mathcal{I}_g$ "Image Embeddings" represent the corresponding target image embeddings. The basic image loss function is an MAE (Mean Absolute Error) function that computes distance between $\mathcal{M}_g$ and $\mathcal{I}_g$.

C. Learning From Self-labeling

To increase the number of examples our model can learn from without needing extra ground-truth labels, we apply pretrained base weights trained only on seen classes for self-labeling. To accomplish this, we run class-agnostic inference on each of the training images followed by standard post-processing and NMS on both detection outputs and ground truth labels. Ground truth labels are always prioritized over generated detections in this NMS operation to ensure that no self-label has significant overlap with a ground-truth label. During training, these self-label boxes are used to augment the data and are treated as ground truth labels for the purposes of typical one-stage detector box regression and objectness losses. We find that this allows our model to learn a more general representation of object localization by providing a wider variety of objects to learn from. For classification losses, the previous method of cropping images and applying the CLIP image encoder to the cropped patches is used along with the same loss function. Both self-label and base label examples are weighted equally in this combined loss. For text loss, these labels are not considered as there is no single corresponding class embedding that this diverse label set could be well aligned with. We visualize examples of self-labels in Figure 1 which shows that rare object classes, not included in the COCO class label set can be identified through self-labeling allowing our model to learn from those examples. With $\mathcal{M}_s$ denoting model semantic outputs that positively match with self-label boxes and $\mathcal{I}_s$ representing the corresponding target image embeddings the full image loss function incorporating self-labels can be written as:

$$\mathcal{L}_{\text{image}} = mean\Big(\Big| \begin{bmatrix} \mathcal{M}_g \\ \mathcal{M}_s \end{bmatrix} - \begin{bmatrix} \mathcal{I}_g \\ \mathcal{I}_s \end{bmatrix} \Big|\Big). \tag{2}$$

D. Full Dual Loss function

Combining both text and image alignment losses into a single function with weighting values $\mathcal{W}_t$ and $\mathcal{W}_i$ [2] produces our full loss function written as:

$$\mathcal{L}_{\text{dual}} = \mathcal{W}_t \mathcal{L}_{\text{text}} + \mathcal{W}_i \mathcal{L}_{\text{image}}. \tag{3}$$

[2]While the total loss could be summarized with a single weighting value and an overall class weight, we find that during hyperparameter fitting, using two weight parameters is more optimal.

Compared to previous ZSD works that only learn from text embedding alignment, our method is able to incorporate this dual loss and learn from both text and image embeddings. We find that this allows embeddings of the same class, which were originally all aligned to the same text embedding value, to be refined by the image loss component so that the embedding space of each class is able to gain meaningful intraclass variance. For example different models of the "car" class could be meaningfully aligned based on differences in color or type based on their image embeddings rather than all being aligned to the same text embeddding as is done in most text only approaches. The result of this process is similar to leveraging instance-wise textual descriptions [22], though our method can be applied without requiring extra data.

E. Inference on Unseen classes

During inference, the similarity matrix $\mathbf{z}$ is computed in the same manner as during training, but with reference embeddings $\mathcal{T}$ set equal to the unseen class name embeddings. We also explore modifying typical YOLO post-processing in order to reduce typical bias of seen objects being given high confidence scores during zero-shot detection and generalized zero-shot detection as they tend to have higher objectness scores.

Text embeddings during inference. When analyzing the cosine similarity between image and text embeddings of certain unseen classes, we found that CLIP often embeds a word incorrectly when only an ambiguous class name is used. [3] For example, the unseen class "mouse" embedding was more semantically similar to the animal mouse, and not the computer mouse, the correct version. Prior ZSD methods avoided this issue by selecting the correct word embedding, based on the WordNet [44] synset, from word vectorizers such as word2vec [45], but the CLIP text embedding does not support this approach. Therefore, we sought a simple, automatic method to correct unseen text reference embeddings that could be applied to any prompt class. To this end, during inference we also include the WordNet [44] definition of the class with the prompt "a photo of {class}, {definition}, in the scene." The inclusion of each class definition not only produces more semantically accurate unseen class reference embeddings, but also distanced previously similar classes by providing specificity. For example, cosine distance between unseen classes "bear" and "cat" increased significantly with the addition of definitions allowing for their class APs to improve by reducing incorrect detections between the two similar classes. While this method does provide an overall score increase, not all class APs improve. Still, we benchmark only applying these evenly across all unseen classes, as benchmarking must adhere to these constraints. In the supplemental materials, we further discuss the results of this process.

Post-processing. To generate predictions, we first apply a very low objectness cutoff which rarely eliminates unseen

[3]This limitation was also found by the original CLIP authors who mentioned this in their code repository: https://github.com/openai/CLIP

class examples compared to two stage approaches. For all remaining anchors, we create a confidence value by multiplying their objectness scores by the maximum class confidence value of each prediction. We then apply a cutoff value to these generated confidence scores. After applying NMS, the final confidence score given to these remaining predictions is only the maximum calculated class similarity value and does not factor in objectness which typically favors seen objects. For our final predictions, we apply NMS and limit max detections because our method tends to produce many high confidence predictions.

IV. Experiments

We compare our proposed method with previous methods under the standard zero-shot detection (ZSD) and generalized zero-shot detection (GZSD) settings. In addition, we present ablation studies to isolate the improvements from each component in our proposed method

A. Dataset

We mainly evaluate model performance under the two most common ZSD benchmarks which use the COCO [35] dataset under the 65/15 [28] and 48/17 [1] ZSD class splits. Both training sets are generated by removing images from the COCO 2014 training set that contain any unseen object instances and sampling the validation images that contain at least one unseen image. To test our model's ability to generalize across datasets we also benchmark on the established ZSD class split for ILSVRC [46] proposed in Rahman *et al.* [21] and the Visual Genome [47] class split proposed in Bansal *et al.* [1].[4] To avoid optimizing training hyperparameters and methods on any of the standard ZSD testing sets, we create a ZSD validation set. To create this validation set, we take all training images with unseen annotations from the 65/15 COCO split, the unused part of the original COCO 2014 training set, resulting in a validation set with 20,483 images.

B. Evaluation Protocol

We benchmark the ZSD and GZSD performance on COCO 48/17 split and 65/15 split. In ZSD, only unseen text embeddings are present during inference and the model predicts only unseen object instances. We report mAP@0.5 and Recall@100 at various IOU thresholds. In GZSD, both seen and unseen text embeddings are present in validation and the model predicts both seen and unseen object instances. Additionally, for GZSD we also report the harmonic mean (HM) of seen and unseen mAP@0.5 and Recall@100 at three IOU thresholds. Similar to previous ZSD papers [27], mAP will always refer to mAP@0.5 unless otherwise specified. For the ILSVRC benchmark, we measure ZSD performance using mAP on the 23 unseen classes. For Visual genome, ZSD performance is measured by averaging the Recall@100 scores of 130 unseen classes at three given IOU scores because the Visual Genome dataset is known to have many missing labels due to the large class set. All benchmarking outside of ablation studies use the proposed self-labeling (III-C), dual loss function (III-D), and ZSD tailored post-processing (III-E).

[4]Since the previous state-of-the-art [27] did not benchmark on these two datasets we compare with the best performing previous method for these benchmarks.

TABLE I
ZSD results on two COCO splits. Seen/Unseen refers to the dataset splits.

Method	Seen/Unseen	Recall@100			mAP
		0.4	0.5	0.6	0.5
SB [1]	48/17	34.5	22.2	11.4	0.4
DSES [1]	48/17	40.3	27.2	13.7	0.6
TD [22]	48/17	45.6	34.4	18.2	-
PL [28]	48/17	-	43.6	-	10.1
ZSD-CNN-ohem [23]	48/17	47.8	41.2	34.4	-
Gtnet [24]	48/17	47.4	44.7	35.6	-
DELO [25]	48/17	-	33.6	-	7.6
BLC [26]	48/17	49.7	46.4	41.9	9.9
ZSI [27]	48/17	57.4	53.9	**48.3**	11.4
ZSD-YOLOx	48/17	**61.6**	**55.8**	46.2	**13.4**
PL [28]	65/15	-	37.8	-	12.4
BLC [26]	65/15	54.2	51.7	47.9	13.1
ZSI [27]	65/15	61.9	58.9	54.4	13.6
ZSD-YOLOx	65/15	**75.5**	**69.5**	**57.3**	**18.3**

C. Implementation Details

Our code implementation is based upon the Pytorch YOLOv5 repository by ultralytics.[5] For model architecture, modifications are made to the final convolutional prediction layer to change class outputs to match the 512 CLIP embedding outputs from the publicly available ViT-B/32 model.[6] For loss we only alter class side losses. To fit hyperparameters, we run hyperparameter search with the COCO 65/15 split training set and use the unseen mAP of our proposed validation set from IV-A as our fitness objective for approximately 200 evolution generations. Our main loss hyperparameters are a text loss weight of 1.05, image loss weight of 1.21, box loss weight of 0.03, class loss weight of 0.469, and objectness loss weight of 2.69. Our main SGD optimizer hyperparameters are base lr of 0.00282, momentum of 0.854, weight decay of 0.00038. Our models are trained on a 50 epoch cosine schedule and initialized with pretrained weights described in Section III.

D. Comparison with State-of-the-art

COCO. We compare our results under the ZSD task for the two COCO ZSD splits in Table I. We observe that under the 48/17 split, our method improves over the previous state-of-the-art approach [27] by 2.0 mAP, a 17.6% improvement. Under the 65/15 split our method surpasses the best scoring previous ZSD method [27] by 4.7 mAP or approximately a 34.6% improvement. Generally, our method performs better under the 65/15 split mainly due because our model is able to approximate the CLIP embedding space better with a more diverse training label set.

[5]https://github.com/ultralytics/yolov5
[6]https://github.com/openai/CLIP

TABLE II
COMPARISON OF OUR METHOD WITH THE PREVIOUS GZSD METHODS ON TWO COCO SPLITS. HM DENOTES THE HARMONIC MEAN FOR SEEN AND UNSEEN CLASS SPLIT MAP AND RECALL@100.

Method	Seen/Unseen	Seen		Unseen		HM	
		mAP	Recall@100	mAP	Recall@100	mAP	Recall@100
DSES [1]	48/17	-	15.1	-	15.4	-	15.2
PL [28]	48/17	36.0	38.3	4.2	26.4	7.4	31.2
BLC [26]	48/17	42.2	57.6	4.6	46.4	8.3	51.4
ZSI [27]	48/17	**46.6**	**70.8**	4.9	**53.9**	8.8	**61.2**
ZSD-YOLOv5x	48/17	31.7	63.3	**13.6**	45.2	**19.0**	52.7
PL [28]	65/15	34.1	36.4	12.5	37.2	18.2	36.8
BLC [26]	65/15	36.1	56.4	13.2	51.65	19.3	54.0
ZSI [27]	65/15	**38.7**	**67.2**	13.7	59.0	20.2	62.8
ZSD-YOLOv5x	65/15	31.7	61.0	**17.9**	**65.2**	**22.9**	**63.0**

Our results for the GZSD COCO task are listed in Table II. Under the GZSD setting, we focus our method development on Unseen mAP and HM (Harmonic Mean) mAP and our method surpasses the previous state-of-the-art [27] in both metrics and under both splits. In the 48/17 split our method improves by 8.7% mAP, or 177.6% under unseen mAP and 10.2 mAP or 116.0% for HM mAP, the main GZSD benchmark. In the 65/15 split our method improves by 4.2 unseen mAP, or a 30.7% improvement. For Harmonic Mean (HM) mAP, our method improves by 2.7 mAP or a 13.4% improvement. Through ablation experiments (IV-F), we find that a large factor in our strong GZSD benchmark scores is due to our proposed post-processing function (III-E).

ILSVRC. Under the ILSVRC split, shown in Table III, many other methods such as DSES [1], ZSDTD [23], and GtNet [24] use additional data or information not present in typical ZSD benchmarking. Despite these advantages that our method does not leverage, we still surpass the previous state-of-the-art by 1.8 mAP or approximately a 6.9% improvement.

Visual Genome. Unlike the COCO and ILSVRC dataset splits, Visual Genome, shown in Table IV is benchmarked only using Recall@100 for three given IOU values because the large class number leads to many missing labels. Our method significantly improves over previous ones. On Recall@100 for IOU=0.5 our method improves over the previous state-of-the-art by 11.7, more than doubling the result [24]. This benchmark shows that our method can improve over other methods far more significantly on larger datasets with a greater number of seen classes to learn from. Additionally, the Visual Genome benchmark shows the strength of our post-processing approach which tends to recall unseen objects far better than typical post-processing as it does not eliminate as many unseen objects through objectness score cutoffs.

E. Inference Speed Comparison

Benchmark settings. Model speeds are evaluated on both single image throughput and mini-batch throughput with batch size determined by the GPU memory utilization of each model. We measure all latencies on the 65/15 dataset split with a single V100 GPU as the listed ZSD approaches have negligible inference time differences between splits. The ZSD-YOLOv5l model is missing from our scaling experiments for reasons explained in our self-labeling ablation study. We also implement our method using the YOLOv3 base detector, and list it as ZSD-YOLOv3. In evaluating inference latency across all models, we include only the amount of time it takes for the model to produce outputs on the batch which includes the calculations of cosine similarities and the softmax with temperature operation, but does not include post-processing operations.

Results. We display the results of our model scaling and compare it to the previous state-of-the-art approach [27], in Table VI. Compared to the previous state-of-the-art [27], our best model, ZSD-YOLOv5x, is over 3x faster on single image inference while also being 4.7 mAP better. For a closer model strength comparison, we train a YOLOv3 model with our method as the strength of the base YOLOv3 model is most similar to the base detector used in ZSI [27] and other ZSD methods and find that our method is still able to outperform ZSI by 3.2 mAP on the 65/15 dataset split while running approximately 9.7x faster on single image inference. To measure scaling performance, we focus on the 48/17 split as our experimentation in developing our method under the 65/15 split likely resulted in certain model scales being slightly more favored in the specific split. In supervised YOLOv5 detection, YOLOv5m and YOLOv5x improve over YOLOv5s by 13.9% and 24.2% respectively in terms of mAP [12] whereas our proposed ZSD-YOLOv5m and ZSD-YOLOv5x improve over ZSD-YOLOv5s by 14.0% and 34.0% in terms of mAP. Given that our relative mAP improvement percentages from scaling are similar to that of the original YOLOv5 models, we conclude that our method scales similarly to fully supervised YOLOv5 detectors.

F. Ablation Study

We present the effects of our proposed self-labeling and post-processing method in Table V, and study the effect of our image loss component seperately in Table VII. In developing our method, we follow previous ZSD literature which uses the unseen mAPs for both ZSD and GZSD as the primary benchmarks and we find using both self-labeling and our proposed post-processing yields the best results across benchmark splits and tasks.

Self-labeling.

Comparing the third and fourth rows of each COCO ZSD split in Table V, we find that self-labeling provides an overall improvement of 0.8 mAP and 1.7 mAP for the 48/17 and 65/15 class splits respectively. Through experimentation, we find that a model which labels using its own base weights tends to create self-labels it has already learned through pretraining and therefore does not "expand" its knowledge of what can be considered an object. For example, we find that when the ZSD-YOLOv5x sized model is self-labeled with ZSD-YOLOv5l, the mAP gain is 0.6 mAP and 0.8 mAP greater, for the 48/17 and 65/15 class splits respectively, than when self-labeled with its own base weights. Therefore, we self-label using the ZSD-YOLOv5l sized detector in our ablation studies and scaling

TABLE III
ZSD CLASS AP FOR UNSEEN CLASSES OF ILSVRC DET 2017 DATASET.

	mean	p.box	syringe	harmonica	maraca	burrito	pineapple	electric-fan	iPod	dishwasher	canopener	plate-rack	bench	bowtie	s.trunk	scorpion	snail	hamster	tiger	ray	train	unicycle	golfball	h.bar
SAN [21]	16.4	5.6	1.0	0.1	0.0	27.8	1.7	1.5	1.6	7.2	2.2	0.0	4.1	5.3	**26.7**	65.6	4.0	47.3	71.5	21.5	51.1	3.7	26.2	1.2
DSES [1]	22.7	7.4	2.3	1.1	0.6	46.2	4.3	8.7	32.7	14.6	6.9	9.1	7.4	4.9	6.9	73.4	7.8	56.8	80.8	24.5	59.9	25.4	33.1	7.6
ZSDTD [22]	24.1	7.8	3.1	1.9	1.1	49.4	4.0	9.4	35.2	14.2	8.1	10.6	9.0	**5.5**	8.1	**73.5**	8.6	57.9	**82.3**	**26.9**	61.5	24.9	38.2	8.9
GTNet [24]	26.0	4.4	**30.4**	2.3	1.2	**51.5**	5.2	18.5	**40.6**	18.0	13.1	4.7	**13.7**	4.6	19.2	69.7	10.2	74.7	72.7	1.4	**65.7**	27.1	**40.4**	**9.1**
ZSD-YOLOv5x	**27.8**	**24.0**	24.4	**6.4**	**2.2**	45.1	**28.1**	**32.0**	24.5	**26.2**	**26.0**	**13.1**	11.7	4.9	17.3	72.9	**33.3**	**78.1**	57.9	0.3	37.3	**37.6**	32.8	4.0

TABLE IV
RECALL@100 COMPARISON OF OUR METHOD WITH THE PREVIOUS STATE-OF-THE-ART ZSD WORKS ON VISUAL GENOME [47].

Method	Recall@100		
	0.4	0.5	0.6
SAN [21]	6.8	5.9	3.1
LAB [1]	8.4	5.4	2.7
ZSDTD [22]	9.7	7.2	4.2
ZSD-CNN-ohem [23]	13.7	11.0	8.3
GTNet [24]	14.3	11.3	8.9
ZSD-YOLOv5x	**27.8**	**23.0**	**15.1**

experiments where we do not test the ZSD-YOLOv5l model (IV-E).

Post-processing. When studying the effect of our post-processing function, we consider row two of each ZSD split in Table V to be the baseline which only uses typical YOLOv5 ZSD post-processing, whereas row four uses our proposed post-processing. We find that our proposed post-processing provides a mAP improvement of 1.6 mAP and 3.6 mAP on the 48/17 and 65/15 COCO ZSD splits respectively. Though when benchmarking GZSD, we find that while our method provides a similar significant unseen class improvement, we also find a loss in seen class detection mAP which is expected as our post-processing method favors unseen classes by reducing the influence of objectness scores when computing detection confidence. Therefore, since our proposed processing has a negligible impact on inference latency, we use our proposed post-processing throughout all of our other experiments. The strength of this post-processing is also another reason we opted for a one stage detection approach as a two stage detection approach cannot consider the class confidences of every anchor due to computational constraints and must first filter out anchors proposals with low objectness scores before computing class scores which we find leads to filtering out unseen object instances.

Image loss component. We study the effect of the image loss component of our loss function by training with image loss suppressed and other hyperparameters the same as the dual loss training. Our results are compiled in Table VII. We find that combining the image loss with the text loss using our proposed dual loss function (ZSD-YOLOv5x-dual) improves ZSD performance by approximately 4.4 mAP on the 48/17 split and 1.5 mAP on the 65/15 split over the text-only baseline (ZSD-YOLOv5x-text). Considering the improvement on the 48/17 split is far greater than the improvement on the 65/15 split, we conclude that the effect additional image embedding loss is accentuated in limited data scenarios but is still able to provide a significant improvement even when there is sufficient data.

V. DISCUSSIONS AND CONCLUSION

Limitations. Despite our method showing significant improvements in ZSD, the problem of the ZSD is still far from solved which limits both positive and negative impacts of our work. An issue within ZSD that has been largely ignored by a majority of ZSD papers is that ZSD performance on seen classes is significantly worse than when using a supervised detector. Another limitation of our method is that our method uses the CLIP model which trains on a wide range of images. Therefore it is likely that the CLIP training set could contain images from unseen classes though we believe our method can still be considered zero-shot due to the generalization capability of CLIP and because CLIP is widely accepted as zero-shot in computer vision research. Furthermore previous ZSD works such as [22], [48] have used image captioning data as well while being considered zero-shot. Zheng *et al.* [27] also demonstrates that any ZSD method must rely on some prior semantic knowledge gained through optimizing an embedding model over all classes which we defined in our problem definition. Therefore, our method, and likely all ZSD methods cannot perform ZSD on truly novel class types like those found in medical diagnoses or optical character recognition as they are unable to be accurately mapped to a semantic embedding space.

Ethics. As the topic of ZSD is relatively underdeveloped, we find there are few immediate negative societal impacts as the result of our research. Still, if applied to a sufficiently large enough training set along with further optimization of ZSD methods, it is likely one could build an object detector that could detect nearly any given object with close to supervised learning accuracy. Models such as these could have profound negative impacts as they could analyze bulk surveillance data, and provide advanced monitoring capabilities. Though, similar to CLIP performing poorly on MNIST, we find that unless a ZSD model has data samples similar to a niche data type, the performance is very weak compared to supervised methods. For example in the ILSVRC dataset, our model performs

TABLE V

COCO ZSD SPLIT RESULTS OF OUR ABLATION STUDY ON OUR PROPOSED SELF-LABELING AND POST-PROCESSING METHODS. NOTE THAT THE BASELINE METHOD IN THIS STUDY USES BOTH IMAGE AND TEXT LOSSES AS WE ABLATE THE IMAGE LOSS COMPONENT SEPARATELY IN TABLE VII. ALL RESULTS ARE PRODUCED USING THE ZSD-YOLOV5X SIZED MODEL.

				ZSD				GZSD					
	ZSD	self-labeling	ZSD-POST	Unseen				Seen		Unseen		HM	
				mAP	Recall@100			mAP	Recall@100	mAP	Recall@100	mAP	Recall@100
				0.5	0.4	0.5	0.6	0.5	0.5	0.5	0.5	0.5	0.5
48/17	✓			10.2	57.5	53.2	47.4	29.7	57.9	9.3	43.9	14.2	50.3
	✓	✓		11.8	60.1	55.6	**49.8**	32.4	63.2	12.7	44.3	18.2	52.1
	✓		✓	12.6	60.2	54.9	45.0	**34.6**	**65.6**	11.1	**51.4**	16.8	**57.6**
	✓	✓	✓	**13.4**	**61.6**	**55.8**	46.2	31.7	63.3	**13.6**	45.2	**19.0**	52.7
65/15	✓			13.7	69.1	64.8	58.6	30.6	57.2	13.7	62.1	18.9	59.6
	✓	✓		14.7	73.8	69.4	**62.4**	**36.3**	**66.0**	14.7	**68.3**	20.9	**67.2**
	✓		✓	16.6	73.9	67.9	56.7	32.7	61.1	16.6	66.4	22.0	63.6
	✓	✓	✓	**18.3**	**75.5**	**69.5**	57.3	31.7	61.0	**17.9**	65.2	**22.9**	63.0

TABLE VI

COMPARISON OF OUR METHOD'S INFERENCE LATENCY WITH PREVIOUS THE STATE-OF-THE-ART [27] ON 65/15 AND 48/17 COCO SPLITS. BATCH SIZES FOR MINI-BATCH THROUGHPUT ARE LISTED IN PARENTHESES IN COLUMN 3.

Method	Latency (ms)		48/17		65/15	
	single	max (batch size)	mAP	Recall	mAP	Recall
ZSI [27]	162.0	100.1 (4)	11.6	54.9	13.6	58.9
ZSD-YOLOv3	16.7	4.6 (20)	11.2	54.1	16.8	66.1
ZSD-YOLOv5s	15.5	1.7 (64)	10.0	47.9	14.0	60.4
ZSD-YOLOv5m	19.1	2.7 (32)	11.4	53.2	17.4	65.9
ZSD-YOLOv5x	51.4	6.4 (16)	13.4	55.8	18.3	69.5

TABLE VII

COMPARISON OF IMAGE LOSS COMPONENT OF OUR DESCRIBED DUAL LOSS FUNCTION WHEN USED TO TRAIN ZSD-YOLOV5X ON THE 65/15 AND 48/17 COCO DATASET SPLITS. WE FIND THAT USING BOTH IMAGE AND TEXT LOSSES (DUAL LOSS) PROVIDES OPTIMAL RESULTS.

Method	Seen/Unseen	Recall@100			mAP
		0.4	0.5	0.6	0.5
ZSD-YOLOv5x-text	48/17	58.8	53.4	45.0	9.0
ZSD-YOLOv5x-dual	48/17	**61.6**	**55.8**	**46.2**	**13.4**
ZSD-YOLOv5x-text	65/15	71.4	66.0	56.4	16.8
ZSD-YOLOv5x-dual	65/15	**75.5**	**69.5**	**57.3**	**18.3**

poorly on the "ray" class, and as shown by [24] there are relatively few classes similar to "ray" in the training set. Therefore, the capabilities of even a generally trained detector can be limited as long as researchers are careful with the datasets they train their models on.

Conclusion. We introduce ZSD-YOLO, a zero-shot detector that surpasses all previous ZSD results on the main ZSD benchmarks using the COCO [35], ILSVRC [46], and Visual Genome [47] datasets. Furthermore, we devise a self-labeling method that can improve ZSD performance without needing new data or labels and optimize single stage post-processing for the ZSD task. We explore the effect of traditional scaling approaches on the ZSD task and find that typical model scaling can transfer well to our method to create a family of efficient and accurate ZSD models. ZSD is an emerging research direction, having many potential applications in computer vision and could lead to a generalized object detector. We hope that our work can establish a strong ZSD baseline that utilizes image and text loss components to inspire further work in the crucial ZSD domain.

REFERENCES

[1] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran, "Zero-shot object detection," in *ECCV*, 2018.

[2] D. Schofield, A. Nagrani, A. Zisserman, M. Hayashi, T. Matsuzawa, D. Biro, and S. Carvalho, "Chimpanzee face recognition from videos in the wild using deep learning," *Science advances*, vol. 5, no. 9, 2019.

[3] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014, pp. 740–755.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[7] R. Girshick, "Fast r-rcnn," in *ICCV*, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[11] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *CVPR*, 2017.

[12] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022.

[13] A. Gupta, P. Dollar, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *CVPR*, 2019.

[14] J. Tan, G. Zhang, H. Deng, C. Wang, L. Lu, Q. Li, and J. Dai, "1st place solution of lvis challenge 2020: A good box is not a guarantee of a good mask," in *arXiv:2009.01559*, 2020.

[15] S. Kalkowski, C. Schulze, A. Dengel, and D. Borth, "Real-time analysis and visualization of the YFCC100m dataset," in *ACM Multimeda Community-Organized Multimodal Mining: Opportunities for Novel Solutions (MMCOMMONS) Workshop*, 2015.

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[17] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale fine-grained visual categorization of birds," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[19] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *IEEE International Conference on Computer Vision Workshops*, 2013.

[20] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," in *arXiv 1306.5151*, 2013.

[21] S. Rahman, S. H. Khan, and F. Porikli, "Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts," in *ACCV*, 2018.

[22] Z. Li, L. Yao, X. Zhang, X. Wang, S. Kanhere, and H. Zhang, "Zero-shot object detection with textual descriptions," in *AAAI*, 2019.

[23] K. Wang, L. Zhang, Y. Tan, J. Zhao, and S. Zhou, "Learning latent semantic attributes for zero-shot object detection," in *ICTAI*, 2020.

[24] S. Zhao, C. Gao, Y. Shao, L. Li, C. Yu, Z. Ji, and et al, "Gtnet: Generative transfer network for zero-shot object detection," in *AAAI*, 2020.

[25] P. Zhu, H. Wang, and V. Saligrama, "Don't even look once: Synthesizing features for zero-shot detection," in *CVPR*, 2020, pp. 11 693–11 702.

[26] Y. Zheng, R. Huang, C. Han, X. Huang, and L. Cui, "Background learnable cascade for zero-shot object detection," in *ACCV*, 2020.

[27] Y. Zheng, J. Wu, Y. Qin, F. Zhang, and L. Cui, "Zero-shot instances segmentation," in *CVPR*, 2021.

[28] S. Rahman, S. Khan, and N. Barne, "Improved visual-semantic alignment for zero-shot object detection," in *AAAI*, 2020.

[29] X. Y, L. C. H, S. B, and A. Z., "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE TPAMI*, 2019.

[30] W. W, Z. V. W, Y. H, and M. C., "A survey of zero-shot learning: Settings, methods, and applications," *ACM Trans. Intell. Syst. Technol*, 2019.

[31] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res*, vol. 2, pp. 263–286, 1995.

[32] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, 2009.

[33] M. Palatucci, D. Pomerleau, and G. E. Hinton, "Zero-shot learning with semantic output codes," in *NeurIPS*, 2009.

[34] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR*, 2009.

[35] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[36] P. Zhu, H. Wang, and V. Saligrama, "Zero shot detection," *IEEE Trans. Circuits Syst. Video Technol*, vol. 30, no. 4, pp. 998–1010, 2018.

[37] ——, "Dont even look once: Synthesizing features for zero-shot detection," in *CVPR*, 2020.

[38] N. Hayat, M. Hayat, S. Rahman, S. Khan, S. W. Zamir, and F. S. Khan, "Synthesizing the unseen for zero-shot object detection," in *ACCV*, 2020.

[39] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang, "Open-vocabulary object detection using captions," in *CVPR*, 2021.

[40] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," 2021.

[41] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[42] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," in *arXiv:1905.05055*, 2019.

[43] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," in *arXiv:1804.02767*, 2018.

[44] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, 1995.

[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[47] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalanditis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *IJCV*, vol. 123, p. 32–73, 2017.

[48] L. Zhang, X. Wang, L. Yao, L. Wu, and F. Zheng, "Zero-shot object detection via learning an embedding from semantic space to visual space," in *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}*. International Joint Conferences on Artificial Intelligence Organization, 2020.

[49] H. Bilen and A. Vedaldi, "Weakly supervised deep detection networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2846–2854.

[50] F. Wan, C. Liu, W. Ke, X. Ji, J. Jiao, and Q. Ye, "C-mil: Continuation multiple instance learning for weakly supervised object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2199–2208.

[51] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 189–203, 2016.

[52] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[53] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *CoRR*, vol. abs/2109.01134, 2021. [Online]. Available: https://arxiv.org/abs/2109.01134

APPENDIX

A. Supplementary Introduction

In the supplementary materials we provide the following items:

1) Information on how to download an anonymized version of our code repository in Section B.
2) Clarification on the differences between zero-shot detection, open-vocabulary detection, and weakly supervised detection and why our method is considered zero-shot detection in Section C.
3) Results of the definition ablation study describing the exact effects of adding definitions to inference text embedding prompts in Section D.
4) Details on our proposed validation set upon which we develop our methods and fit hyperparameters in Section E
5) Discussion of qualitative results demonstrating both the strong performance of our model as well as examples of common mistakes in Section F.
6) More information on implementation details including our hyperparameter search process, post-processing constants, and self-labeling constants in Section G.
7) An additional ablation study on the Visual Genome [47] ZSD split [1] justifying our choice of post-processing in Section H.

B. Code Information

An anonymized copy of our code repository can be obtained at this embedded google drive link(5.1 GB). The code currently supports validation on 65/15 COCO testing set (ZSD, GZSD, scaling exps) as other dataset splits are too large. Weights for all dataset splits can still be found in weights/model_checkpoints folder. Once requirements are installed using the requirements.txt file, the inference commands found within the README.md can be run.

C. Zero-shot Detection vs Open-vocabulary Detection vs Weakly Supervised Detection

Given the similarities between the tasks of zero-shot detection (ZSD), open-vocabulary detection (OVD), and weakly supervised detection (WSD), we would like to clarify the differences and explain why we have classified our method as ZSD. While these tasks all address the issue of learning to detect objects using limited data, each provide different constraints on the type of training data that can be used.

Firstly, weakly supervised detection, arguably the outlier in this suite of tasks, uses only image-labels to train a detector. Therefore, the main challenge in WSD [49]–[51] is localizing novel objects since novel classes are often present in the training data. WSD approaches generally also require knowledge of the target novel classes during training, whereas ZSD and OVD approaches must be able to target any subset of the entire language vocabulary.

In comparing OVD and ZSD, ZSD is the more restrictive task, as acknowledged by Zareian *et al.* [39]. The main difference between the OVD benchmarks and ZSD benchmarks are that ZSD training datasets eliminate all training images with instances of unseen objects whereas OVD benchmarks do not.[7] While the labels of unseen objects are removed, OVD models are still able to learn from these examples. Most recently the method proposed by Gu *et al.* [40] distills vision-language information to detect unseen classes. The method of Gu *et al.* [40] is able to learn from background examples through their proposed image distillation method. Using only text distillation, they achieve only 5.9 mAP[8] [40] on unseen classes whereas using only image distillation they achieve 24.1 mAP [40] on the 48/17 COCO OVD benchmark from [39]. In ablation studies, they demonstrated that the contribution of the image and text losses were approximately the same under their proposed LVIS [13] benchmark. Note that our text only method is still able to achieve 9.0 mAP under the 48/17 ZSD benchmarking split which eliminates all training images with unseen samples. Given these results, we conclude that the inclusion of unseen objects in the background of images provides an enormous advantage to OVD methods. The authors of [40] also acknowledge this fact in discussing their COCO 48/17 OVD [39] results. With this in mind, ZSD methods must be able to perform detection on classes that are not present in training images at all whereas OVD methods require the inclusion of unseen objects within their training sets to perform well. Therefore, when unseen objects are unavailable, as is often the case since one could not simply just include unlabeled images in their object detection training sets, OVD models fail to perform well in detecting those entirely unseen object types.

While Zareian *et al.* [39] states the usage of image captioning data classifies a method as OVD, previously published works [22], [48] have used similar image captioning while still being classified as ZSD. For this reason, we believe the usage of CLIP [16], which is trained on a large corpus of image text pairs can still be considered a ZSD method. Additionally, CLIP has been considered a zero-shot model since its introduction and in the various downstream tasks leveraging its embedding space. Furthermore, the CLIP embedding space is optimized for a vocabulary of similar size compared to other embedding spaces used such as GloVe [52] and word2vec [45]. Therefore, in terms of application, our method can transfer to a similar number of novel class names. The CLIP encoders also support full sentence prompts as demonstrated by our inference prompts including definitions allowing for nearly any prompt to be used for detection, thus our method can be applied just as widely, if not more widely than previous ZSD methods. While we concede that the CLIP embedding space may have stronger visual-semantic mappings, we cannot apply our method using previous word vectorizers to directly compare with their results. Similarly, we are unsure how to apply previous methods using CLIP embeddings and therefore cannot in good faith attempt to implement their methods to

[7] With the exception of the Visual Genome ZSD benchmark for reasons explained in [1]

[8] Similar to our main paper, mAP refers to mAP@0.5

TABLE VIII

ABLATION STUDY CLASS-WISE AP FOR UNSEEN CLASSES OF COCO 65/15 ZSD SPLIT [28]. ZSD-YOLOV5X DENOTES NOT USING DEFINITIONS, ZSD-YOLOV5X + DEFINITION DENOTES USING DEFINITIONS. NOTE THAT THE MEAN FOR GAIN PERCENTAGE IS THE ACTUAL MEAN OF THE GAIN PERCENTAGES, NOT THE PERCENT INCREASE IN MAP.

	mean	tie	airplane	elephant	dog	knife	sink	bus	snowboard	cat	cake	cow	cup	scissors	couch	keyboard	skateboard	umbrella
ZSD-YOLOv5x	13.0	0.0	10.5	18.8	11.3	1.5	3.3	51.6	21.6	5.3	5.7	34.9	13.3	0.1	31.9	8.2	0.6	2.6
ZSD-YOLOv5x + definition	13.4	0.0	19.0	29.6	16.7	2.1	4.5	62.9	24.1	5.8	4.9	27.9	10.4	0.1	15.5	3.7	0.2	0.1
Gain	+0.4	0.0	+8.6	+10.8	+5.4	+0.6	+1.1	+11.3	+2.5	+0.5	-0.8	-7.0	-3.0	-0.0	-16.4	-4.5	-0.5	-2.5
Gain Percentage	+47.6%	+865.3%	+81.9%	+57.5%	+48.2%	+42.4%	+34.3%	+22.0%	+11.6%	+9.7%	-13.2%	-20.1%	-22.2%	-33.2%	-51.3%	-54.6%	-73.0%	-95.4

TABLE IX

ABLATION STUDY CLASS-WISE AP FOR UNSEEN CLASSES OF COCO 65/15 ZSD SPLIT [28]. ZSD-YOLOV5X DENOTES NOT USING DEFINITIONS, ZSD-YOLOV5X + DEFINITION DENOTES USING DEFINITIONS. NOTE THAT THE MEAN FOR GAIN PERCENTAGE IS THE ACTUAL MEAN OF THE GAIN PERCENTAGES, NOT THE PERCENT INCREASE IN MAP.

	mean	bear	cat	airplane	parking meter	mouse	toaster	train	hair drier	hot dog	frisbee	snowboard	fork	suitcase	toilet	sandwich
ZSD-YOLOv5x	14.9	19.6	16.5	14.0	1.7	2.9	1.1	28.4	0.5	7.4	28.0	39.2	16.4	11.2	17.6	18.9
ZSD-YOLOv5x + definition	18.3	54.0	36.3	18.3	2.0	3.3	1.2	32.2	0.6	7.9	29.4	39.5	15.1	10.1	12.9	11.8
Gain	+3.4	+34.4	+19.8	+4.3	+0.3	+0.4	+0.1	+3.7	+0.1	+0.5	+1.4	+0.4	-1.3	-1.0	-4.7	-7.0
Gain Percentage	+21.9%	+175.1%	+120.0%	+30.4%	+20.0%	+14.3%	+13.5%	+13.1%	+10.1%	+6.7%	+5.0%	+0.9%	-7.9%	-9.0%	-26.7%	-37.3

draw a comparison.

D. Definition Ablation

To study the direct effects of adding definitions to inference prompts on each class, we present the class-wise APs with and without definitions in inference prompts for the COCO 48/17 [1] and 65/15 [28] ZSD splits in Table VIII and Table IX respectively. Following the analysis in our main paper, the "bear" and "cat" class APs had the largest increase under the 65/15 benchmarking split, increasing by 175.1% and 120.0% respectively, given that their embedding cosine similarities decreased from 0.924 to 0.775 with the addition of definitions. The case of these two classes demonstrates that the inclusion of definitions can aid in differentiating between two previously similar text embeddings and reduce the number of false positives between the two classes. Furthermore, other classes such as the "mouse" class that were previously mapped incorrectly due to the ambiguous nature of some class names benefited from the addition of definitions. Lastly, classes such as the "airplane" or "elephant" class benefited due to size, color, or shape descriptions present in the definitions which provided a more accurate visual-semantic embedding. While many class APs decreased with the addition of definitions, we believe that adding definitions to inference prompts remains a simple and effective way to provide better visual-semantic embeddings for ambigious class names. Furthermore, we believe that the addition of definitions to prompts do not violate ZSD benchmarking rules given that other methods are able to obtain proper word embeddings by simply hand selecting the correct WordNet synset, a method that is not available to CLIP. When applying a ZSD method to real world problems, users will likely have at least one example of their target class to test inference, and therefore can tune their prompts manually or use vision-language prompt engineering methods [53], though we avoided doing so in our paper as ZSD methods cannot prompt tune in this manner.

E. Validation Set Details

While all other ZSD benchmark datasets used in our main paper have been developed in prior works [1], [21], [27], [28], we provide the details of our proposed validation set for future use. The validation set is created by taking the portion of the COCO 2014 training split that is not used in 65/15 ZSD training or testing. This validation set contains 20,483 images with 33,690 unseen box annotations. Our validation set has the same unseen classes as the COCO 65/15 testing set. We compare the class distributions in Figure 2, and find that the two distributions are relatively similar with a mean-normalized chi-square distance of approximately 0.206.

We did not create a validation set for the 48/17 COCO split and other benchmarks because our developed methods were transferred from the 65/15 COCO split. Our class-wise AP results for the COCO 65/15 split and 48/17 split are displayed in Table XII and Table XIII respectively.

F. Qualitative Results

We display a couple examples of our zero-shot detector on the COCO 65/15 testing dataset split in Figure 3. We find our proposed model is able to detect diverse unseen classes under

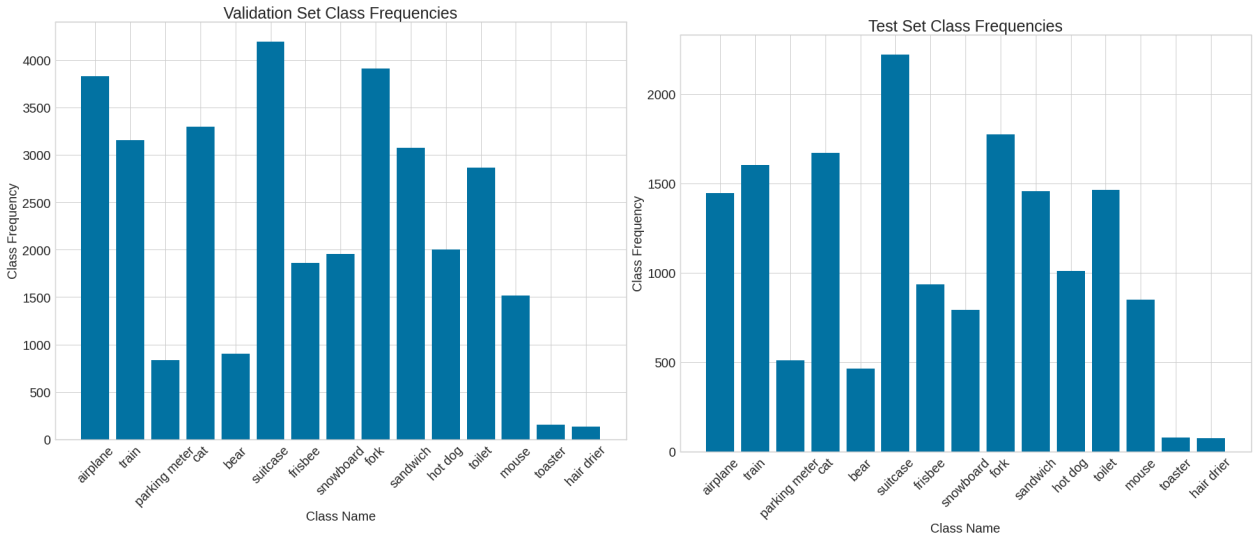

Fig. 2. Histogram comparison of class frequencies between proposed validation set and typical testing set of COCO 65/15 ZSD class split. The validation set is created by taking the portion of the COCO 2014 training split that is not used in 65/15 ZSD training or testing.

various situations with well refined boxes. The top left image shows our model's ability accurately detect the unseen class of "bear" despite two instances being obscured by grass and shadow. The top middle image reveals our model's ability to perform dense zero shot detection as it detects most of the sandwiches while both filtering our similar objects that are not labeled as sandwiches and correctly classifying the hot dog. Lastly the top right image show our model's ability to differentiate between multiple similar and overlapping objects of the "train" class against a noisy background. When examining the mistakes of our model displayed in the bottom row, we mainly see an issue with false positives of seen classes being recognized as unseen classes. In the images shown in the bottom row, we see "keyboard" being detected as "mouse," "person" being detected as "snowboard," and "dog" being detected as "bear." We believe that our softmax based class prediction is a cause for this issue since even when a semantic output is dissimilar from a reference class embedding, if it is relatively close compared to other reference embeddings, it may be assigned a high confidence. To address this issue we experimented with a sigmoid based activation where the class confidences would not be dependent on similarity to other class embeddings. This method did not perform well in training because when applying a "temperature" factor similar to that found in temperatured softmax to sigmoid, a majority of gradients vanish due to extremely low gradients on the left and right sides of the activation. Further research will be needed to work to better filter out objects that are not present in the given embedding set from being detected.

G. Further Implementation Details

a) ***Hyperparameter search.****:* After conducting pretraining as described in our main paper, we fit detection training parameters for our ZSD-YOLOv5x model using mAP on our proposed COCO 65/15 validation set as the fitness parameter. Our final hyperparameter search values are listed in Table X. Across all dataset splits and evaluation benchmarks, the only change made to hyperparameters is a heuristic increase in temperature value for the Visual Genome ZSD benchmark [1] due to the greater number of classes present.

TABLE X
ZSD TRAINING SETTINGS FOR COCO DATASET SPLITS. AUGMENTATIONS ARE SHOWN IN THE SECOND HALF OF THE TABLE AND THEIR LIKELIHOODS ARE INDICATED AS A RATIO BETWEEN 0 AND 1.

config	value
optimizer	SGD
base learning rate	0.00282
weight decay	0.00038
optimizer momentum	0.85403
batch size	20
learning rate schedule	cosine decay
positive iou threshold	0.14671
box loss weight	0.03
class loss weight	0.04688
objectness loss weight	2.67895
temperature	3.91
image rotation (+/- deg)	0.373
image flip left-right (probability)	0.5
image flip up-down (probability)	0.00856
image HSV-Hue augmentation (fraction)	0.0168
image HSV-Saturation augmentation (fraction)	0.7876
image HSV-Value augmentation (fraction)	0.45518
image mosaic augmentation (probability)	0.94109
image scale (+/- gain)	0.64818
image shear (+/- deg)	0.602
image translation (+/- fraction)	0.16587

b) ***Post-processing.****:* For our proposed post-processing function, we use the following parameters. NMS-IOU-threshold = 0.4, initial-confidence-threshold = 0.001, second-

Fig. 3. Selected examples of our results for zero shot detection produced by our ZSD-YOLOv5x model on the COCO 65/15 testing set. Well detected images are in the top row while poorly detected images are displayed in the bottom row. Best view in color.

TABLE XI
COMPARISON OF EFFECT OF PROPOSED POST-PROCESSING ON THE VISUAL GENOME BENCHMARK. YOLO-POST STANDS FOR THE TYPICAL BASELINE POST-PROCESSING, ZSD-POST STANDS FOR OUR PROPOSED POST-PROCESSING, AND ZSD-POST+ STANDS FOR OUR PROPOSED POST-PROCESSING THAT USES OBJECTNESS IN NMS BUT NOT IN THE FINAL CONFIDENCE PREDICTION.

Method	Recall@100			mAP
	0.4	0.5	0.6	0.5
ZSD-YOLOv5x + YOLO-POST	24.6	22.0	**18.5**	1.64
ZSD-YOLOv5x + ZSD-POST	27.8	22.3	15.1	**1.96**
ZSD-YOLOv5x + ZSD-POST+	**28.6**	**24.4**	18.4	1.73

confidence-threshold[9] = 0.1, max-detection = 15. Across datasets and benchmarks certain post-processing parameters change heuristically. For example in GZSD, max-detections increases to 45, and in Recall@100 evaluations max-detections increases to 100. Confidence values also change depending on the benchmark. Full details and paramters are shown in our code implementation attached.

*c) **Self-labeling.**:* For self-labeling experiments we limit self-label boxes to only those which are greater than 25 pixels in both width and height. An initial objectness cutoff of 0.3 is applied followed by IOU with 0.2 threshold to maximize unique boxes. Self-labeling is performed only for training sets which will not be released in the initial supplementary materials due to file size limitations. Still, all weights are trained via self-labeling and are available in the initial attached release.

[9]Second confidence cutoff applied to confidence values that are equal to the product of objectness score and max class similarity

H. Visual Genome Ablation

To further verify the efficacy of our proposed post-processing function we perform an additional ablation experiment on the Visual Genome ZSD benchmark [1] to examine how our post-processing affects both Recall@100 and mAP. We present our results in Table XI. We find similar results to those of our COCO ablation experiments with our proposed post-processing performing better under all benchmarks except for Recall@100 IOU=0.6. We believe this is largely because our method factors objectness score less and therefore lower objectness boxes may be favored over higher objectness boxes that typically match with ground truth box more closely in NMS. To address this issue, we also create another post-processing function denoted as ZSD-POST+ in the table. In this operation, we factor objectness into the NMS value by multiplying the maximum similarity value of each class before NMS then only using maximum similarity class values for final confidence. In comparison, our usual proposed post-processing uses only maximum class values for NMS and final confidence scores. For the Visual Genome benchmark we find that this operation provides the best performance under all listed recall benchmarks. While we also experimented with this function on the original COCO benchmarks, we did not find improvements. For example on the COCO 65/15 benchmark [28] we find that this function lowers mAP from 18.3 to 17.6. Therefore, in our main paper we only use the original post processing function, ZSD-POST, as we must transfer all methods across benchmarks with only minor parameter changes.

TABLE XII

ZSD CLASS AP FOR UNSEEN CLASSES OF COCO 65/15 ZSD SPLIT [28] FOR 4 GIVEN YOLO MODEL VARIANTS. NOTE THAT ALL METHODS USE BOTH OUR ZSD TAILORED POST-PROCESSING FUNCTION AND SELF-LABELING AUGMENTATION. NOTE THAT SCALING IMPROVEMENTS ARE RELATIVELY CONSISTENT ACROSS CLASS APS SHOWING THAT TYPICAL DETECTOR SCALING RULES CAN TRANSFER WELL TO OUR ZSD METHOD.

	mean	airplane	train	parking meter	cat	bear	suitcase	frisbee	snowboard	fork	sandwich	hot dog	toilet	mouse	toaster	hair drier
ZSD-YOLOv5x	18.3	18.3	32.2	2.0	36.3	54.0	10.1	29.4	39.5	15.1	11.8	7.9	12.9	3.3	1.2	0.6
ZSD-YOLOv5m	17.4	23.2	29.3	6.0	32.8	50.9	6.8	27.3	36.8	13.4	10.5	7.3	10.1	5.1	0.9	0.8
ZSD-YOLOv5s	14.0	15.7	24.6	6.2	28.9	46.3	5.9	21.9	29.1	8.6	7.2	3.9	7.1	4.8	0.5	0.0
ZSD-YOLOv3	16.8	20.3	28.3	4.4	30.7	50.9	9.4	26.6	40.9	10.1	9.1	6.8	10.7	2.9	1.0	0.5

TABLE XIII

ZSD CLASS AP FOR UNSEEN CLASSES OF COCO 48/17 ZSD SPLIT [1] FOR 4 GIVEN YOLO MODEL VARIANTS. NOTE THAT ALL METHODS USE BOTH OUR ZSD TAILORED POST-PROCESSING FUNCTION AND SELF-LABELING AUGMENTATION. NOTE THAT SCALING IMPROVEMENTS ARE RELATIVELY CONSISTENT ACROSS CLASS APS SHOWING THAT TYPICAL DETECTOR SCALING RULES CAN TRANSFER WELL TO OUR ZSD METHOD.

	mean	airplane	bus	cat	dog	cow	elephant	umbrella	tie	snowboard	skateboard	cup	knife	cake	couch	keyboard	sink	scissors
ZSD-YOLOv5x	13.4	19.0	62.9	5.8	16.7	27.9	29.6	0.1	0.0	24.1	0.2	10.4	2.1	4.9	15.5	3.7	4.5	0.1
ZSD-YOLOv5m	11.4	13.0	57.4	4.3	20.4	19.7	22.2	0.1	0.0	19.4	0.4	8.1	2.1	6.5	11.9	3.7	4.4	0.2
ZSD-YOLOv5s	10.0	12.1	55.3	4.6	14.9	16.7	17.6	0.1	0.0	17.8	0.3	7.3	3.1	4.4	10.5	1.6	3.6	0.1
ZSD-YOLOv3	11.2	9.3	58.0	4.7	16.3	23.3	23.4	0.2	0.0	20.3	0.2	7.7	2.3	4.0	12.4	2.4	5.8	0.1