One/Zero Shot Vehicle Detection on Satellite Images using Image/Text Queries

Pelinsu Acar

10/11/2024

1. INTRODUCTION

The goal of this study is to implement and compare the performances of two zero shot object detection models which are OWL-ViT and YOLO-World. Both models are tested using image and text queries. While OWL-ViT offers text-conditioned object detection in zero-shot manner, the model has also the capability to detect objects using image-queries in one shot manner. On the other hand, YOLO-World can utilize both text and image queries as in zero shot object detection. Since OWL-ViT is considered as baseline, first the two models are experimented using pre-trained weights. After that YOLO-World is trained to increase the performance further for both text queries and image queries. For the performance comparision both f1 and AP scores are considered to reveal the capability of two models using image and text queries.

2. DATASET ANALYSIS

The "Vehicle Detection from Satellite" dataset sourced from Roboflow [1], is used for this project. It is designed to support object detection tasks, focusing on vehicles as seen from satellite imagery. Each image contains annotations for vehicles across varied environments, including urban and rural settings. The dataset offers diverse perspectives, lighting, and spatial contexts, aiding in the development of robust vehicle detection models in satellite images.

It contains 2,059 images annotated in COCO [2] format with 6,554 annotated vehicle instances across 10 classes, averaging 3.2 objects per image. Due to class imbalance, only the classes *car*, *van*, *truck*, *pickup*, and *camping car* were used for the experimentation. Moreover, to reduce the class imbalance, some images that contains only car are excluded. The final dataset contains 501 images with 1761 annotated vehicle instances across 5 classes. The more detail about the class distribution is illustrated in figure 1.

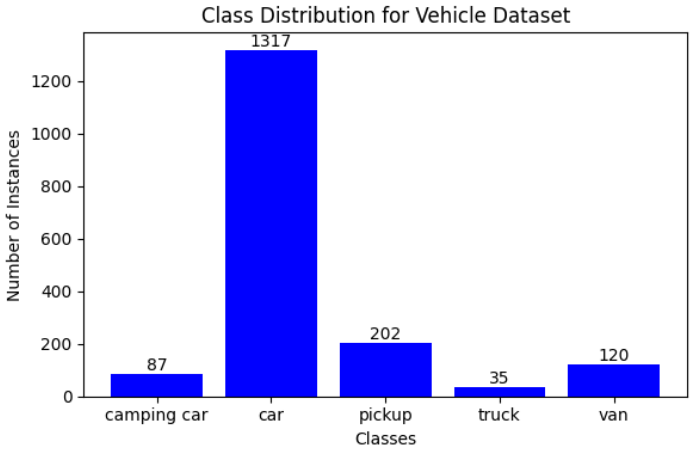

Figure 1: Class distribution of the filtered vehicle dataset

Further analysis about the number of objects per image for each category reveals that most images depict cars, with an average of 2.63 cars per image, while pickup, camping car, van and trucks appear at 1.15, 1.14, 1.25 and 1.09 per image on average respectively.

In an analysis of object area proportions within images, the class *truck* occupies the highest average proportion (0.08), while other classes remain under 0.03. The illustrations of histogram analysis can be found in Appendix A. The class imbalance and varying object sizes provide both challenges and insights, making this dataset suitable for studying object detection performance under these conditions.

3. DESCRIPTION OF MODELS

In this project, I implemented two OVOD models for the kernel detection task, each with distinct methodologies: YOLO-World [3] is used as a zero-shot object detection model by feeding it semantic embeddings of query images and query texts, while OWL-ViT [4] operates as both zero-shot detector utilizing embeddings of text queries and one-shot open-set detector capable of taking query images directly as input. This setup allows YOLO-World to generalize to novel objects based on learned embeddings without seeing any instances at inference, whereas OWL-ViT detects objects by visually comparing them to a single provided example image. Through this approach, this project explores each model's strengths and limitations in handling zero-shot and one-shot detection tasks, comparing their performances for both image and text queries.

3.1 OWL-ViT [4]

OWL-ViT (Open-Vocabulary Learning with Vision Transformers) is an object detection model designed to recognize a wide range of objects beyond the specific categories it was trained on. The architecture consists of a vision transformer encoder and a detection head.

OWL-ViT uses a standard Vision Transformer [5] as the image encoder and a similar Transformer architecture as the text encoder. The image encoder is used to encode the input image into token representations. To adapt the image encoder for detection, it removes the token pooling and final projection layer, and instead linearly project each output token representation to obtain per-object image embeddings for classification. This means that each token representation corresponds to an object in the image. The maximum number of predicted objects is therefore equal to the number of tokens (sequence length) of the image encoder.

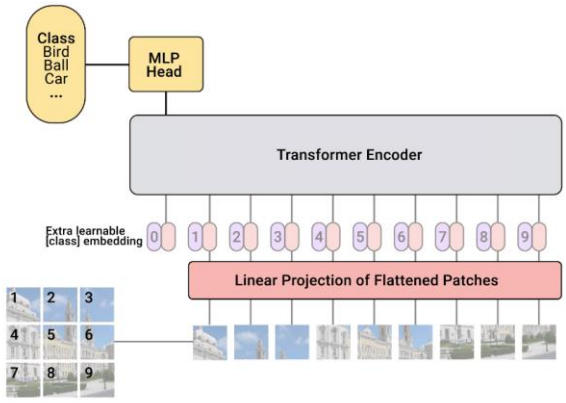

Figure 2: Architecture of Vision Transformer (ViT) [5]

As it is illustrated in figure 2, the maximum number of predicted objects can be found by this formula: $\frac{Input\ Size^2}{Patch\ Size}$

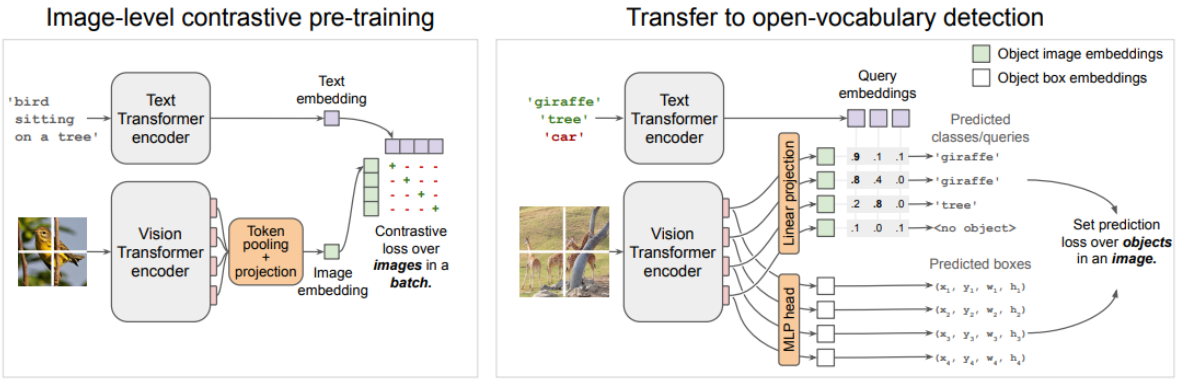

Figure 3: Overview of OWL-ViT published in the original paper [4]

Image and text models are first pre-trained on the image level using contrastive loss, which is a metric learning loss function as it calculates the Euclidean distance or cosine similarity between vector pairs, and then finetuned on object-level annotations. After pre-training, token pooling is removed to preserve the spatial information of individual tokens for tasks such as object localization and detection heads are added. The detection-specific heads contain at most 1.1% (depending on the model size) of the parameters of the model. Box coordinates are obtained by passing token representations through a small MLP.

OWL-ViT includes no fusion between image and text encoders. Therefore, this setup does not require query embeddings to be of textual origin. Since there is no fusion between image and text encoders, we can supply image- instead of text-derived embeddings as queries to the classification head without modifying the model to perform image-conditioned one-shot object detection. Using image embeddings as queries allows detection of objects which would be hard to describe in text.

3.2 YOLO-WORLD [3]

YOLO-World's architecture consists of three key elements: a YOLO detector, a Text/Image Encoder for query objects, and a Re-parameterizable Vision-Language Path Aggregation Network (RepVL-PAN). The overall architecture can be seen in figure 4.

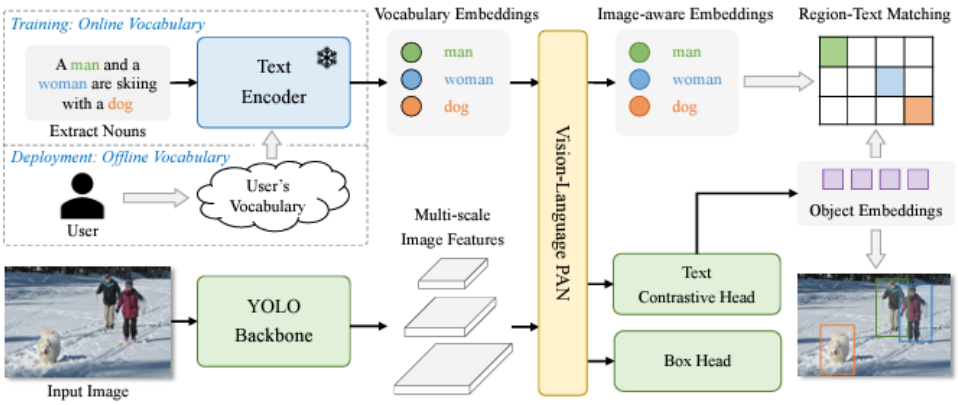

Figure 4: Architecture of YOLO-World published in original paper [3]

In the following sections, each component of YOLO-World is explained in detail.

3.2.1 Backbone

YOLO-World, developed based on YOLOv8 [6], utilizes a CSPDarknet53 [7] backbone for multi-scale feature extraction. The Cross Stage Partial (CSP) [7] architecture splits the feature map into two parts, enhancing learning and reducing computation through convolution operations and concatenation. YOLOv8 introduces the C2f module, combining the C3 module with ELAN from YOLOv7 [8], featuring ConvModules and DarknetBottleNecks to enrich gradient flow. The SPPF (Spatial Pyramid Pooling - Fast) layer enables multi-scale feature representation, crucial for detecting objects of varying sizes. This optimized SPP [9] variant maintains spatial dimensions with fewer resources by using max-pooling layers at different scales. The backbone produces x number of grids per image and then produce one object prediction per grid cell since yolov8 does not rely on any anchor boxes. The number of grids depend on the input image size. An example of number of grid calculation considering the image size in our dataset is defined as follows:

Input Size	Feature Map Strides	Number of Grids	Total Grids
640 x 640	8	$(6400/8)^2 = 6400$	8400
	16	$(6400/16)^2 = 1600$	
	32	$(6400/32)^2 = 400$	

Table 1: Number of grids calculation for a given input size

3.2.2 Re-parameterizable Vision-Language Path Aggregation Network (RepVL-PAN)

RepVL-PAN performs multi-level cross-modality fusion between image features and text embeddings. The fusion between image features and text embeddings is implemented via Text-guided Cross Stage Partial Layer (T-CSPLayer), built on top of the C2f layer, used in the YOLOv8 architecture, by adding text guidance into multi-scale image features. This is achieved through the Max Sigmoid Attention Block, which computes attention weights based on the interaction between text guidance and spatial features of the image. These weights are then applied to modulate the feature maps, enabling the network to focus more on areas relevant to the text descriptions.

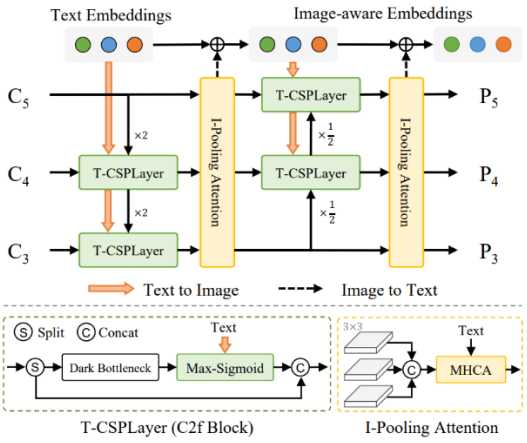

Figure 5: Architecture of YOLO-World neck published in the original paper [3]

3.3.3 Constrastive Head

Similar to OWL-ViT**, t**he contrastive head of yolo-world obtains object-text similarity using L2 norm. In addition, it contains some affine transformation with the learnable scaling factor α and shifting factor β. Both the L2 norms and the affine transformations are important for stabilizing the region-text training.

4.0EXPERIMENTAL SETUP

4.1 Metrics

For the evaluation of our models, both mAP that calculates the average precision across all classes and AP per each class are considered. Furthermore, since these models produces thousands of bounding boxes for each image, the f1 scores and number of different type of errors at various confidence thresholds are analyzed to decide the best confidence threshold and filter out the irrelevant predicted boxes.

4.2 OWL-ViT

Different architectures of the OWL-ViT visual transformer is illustrated in table 2. In this project, ViT-Base model with 16×16 input patch size is used.

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 2: Different ViT models parameters

Therefore, the input image is split info square patches of shape (p, p, c) = (16,16,3), where p = patch-size. The number of patches can be calculated from the formula explained in section 3.1. Since our model uses inputs with size 960x960, there will be 3600 square patches.

These patches are flattened, resulting in n=3600 line vectors of shape (1, p^2*c) = (1,768). The flattened patches are multiplied with a trainable embed ding tensor of shape (p^2*c, d) = (768,512), which learns to linearly project each flat patch to dimension d. This dimension d is constant in the architecture and it is defined in figure xx. Therefore, owl-vit model generates 3600 embeddings with shape (1,512) per image and multiply them with the query embeddings to match them correctly. Both the processor and the object detection model is initialized from the pretrained checkpoint **"google/owlv2-base-patch16-ensemble".** The summary of the OWL-ViT architecture can be seen in table 3.

Component	Parameter/Setting	Details
Input	`image_size`	960 (dimensions of input image: 960x960)
	`patch_size`	16 (size of each image patch: 16x16)
Vision Transformer	`vision_hidden_size`	768 (hidden state size for vision model)
	`vision_intermediate_size`	3072 (size of the intermediate layer)
	`vision_num_hidden_layers`	12 (number of transformer layers)
	`vision_num_attention_heads`	12 (number of attention heads per layer)
Projection Layer	`projection_dim`	512 (dimension for projected features)
Text Transformer	`text_hidden_size`	512 (hidden state size for text model)
	`text_intermediate_size`	2048 (size of the intermediate layer)
	`text_num_hidden_layers`	12 (number of transformer layers)
	`text_num_attention_heads`	8 (number of attention heads per layer)

Table 3: Architecture details of OWL-ViT

4.3 YOLO-World

4.3.1 Dataset Splitting

For training and evaluation of the YOLO-World model, the dataset was split to assess both traditional training performance and zero-shot learning capabilities. The dataset includes five vehicle classes: “camping car,” “pickup,” “truck,” “car,” and “van.” Out of these, three classes (“camping car,” “pickup,” and “truck”) were selected for training, while the remaining two classes (“car” and “van”) were reserved exclusively for zero-shot testing. This ensures that the model’s ability to detect new, unseen classes can be properly evaluated.

To create the training and test sets for the “camping car,” “pickup,” and “truck” classes, 10 images from each class were set aside for the test set. The remaining images for these three classes were used for training. Since some classes are underrepresented, small number of images are preserved for the test set to have enough data for the training. This method provides a balanced approach where the model is trained on most of the available data while still being evaluated on unseen examples.

4.3.2 Training of YOLO-World

Instead of train the model from scratch, the model is initialized using the pre-trained version which was trained on COCO dataset for 40 epochs with a learning rate of 1e-3. The training of the YOLO-World model was configured using a custom adaptation of the YOLOv8 large model as a base. The training was set for 50 epochs not to lose the zero-shot performance, with an initial learning rate of 2e-3 and a weight decay of 0.0005 to prevent overfitting. Stochastic Gradient Descent (SGD) was used as an optimizer with momentum of 0.937. The batch size was configured at 4 images per GPU. Mosaic augmentation was used only for the first 20 epochs to enhance data variability while ensuring natural training data distribution in later epochs. However, in the experiments it was observed that the best balance between standard object detection and zero-shot performance is obtained at epoch 30. Therefore, for all the evaluations, the training weights at 30. epoch are used. The summary of the architecture and training settings of the model can be seen in table 4.

Component	Parameter/Setting	Details
Text Embedding	`text_channels`	512 (dimensionality of text embeddings)
Neck	`neck_embed_channels`	[128, 256, `_base_.last_stage_out_channels // 2`]
		Embedding channels for each stage of the neck
	`neck_num_heads`	[4, 8, `_base_.last_stage_out_channels // 2 // 32`]
		Number of attention heads for each neck stage
Optimizer	Type	SGD
	Learning Rate (`lr`)	2e-3
	Momentum	0.937
	Nesterov	True
	Weight Decay	0.0005
	Batch Size per GPU	4

Table 4: Architecture details of YOLO-World

In terms of loss functions, YOLO-World utilized a combination of object detection loss components. The primary losses included the **bounding box regression loss** to refine localization accuracy, **classification loss** to ensure correct class predictions, and **objectness loss** to determine the presence of objects within proposed regions. These were augmented with specialized modules, like focal sigmoid

loss for improved handling of class imbalance, and bipartite matching strategies to ensure optimal label assignments during training. This combination of loss functions allowed YOLO-World to effectively align detected bounding boxes with query embeddings, ensuring accurate prediction for both seen and unseen classes.

4.3.3 Training CLIP Embeddings

CLIP embeddings, which are used as queries for your YOLO-World model, sometimes fail to effectively distinguish between objects of the same class. This can result in poor object detection performance. By fine-tuning CLIP [10], the model learns to better cluster embeddings of similar objects (target and related queries) together, improving the ability of the model to differentiate between objects during inference. The model is initialized from the pre-trained checkpoint **"openai/clip-vit-base-patch32".** The training setup involves fine-tuning the CLIP model's vision component using **TripletMarginLoss** with a margin of 0.5, defining the minimum separation that the model should maintain between dissimilar pairs in the embedding space, to improve the clustering of similar object embeddings. The training was set for 30 epochs with an initial learning rate of 5e-6, and batch size of 5. The **MultiSimilarityMiner** is used to identify hard positive and negative pairs, guiding the model to better separate related query embeddings. The optimizer is Adam, with a **Cosine Annealing scheduler** to adjust the learning rate, and gradient clipping with a value of 0.7 ensures stable training. The result is evaluated by plotting the embeddings of query and target images on a t-sne plot.

5.0 RESULTS

5.1 OWL-ViT

For the OWL-ViT performance with image queries, figure 6 shows a steady decline in mAP as the IoU threshold increases from 0.5 to 0.9 while the highest mAP of 0.31 is observed at an IoU of 0.5**.** The model performs well at lower IoU thresholds (e.g., 0.5–0.6), indicating that it can localize objects with a moderate level of accuracy and struggles with precise localization. Figure 8 for text queries shows that, the initial mAP is slightly higher at 0.34 at an IoU of 0.5 than for image queries, but it follows a similar downward trend, with a sharp decline to near zero at 0.9.

From figure 7 for image queries shows that “Car” category has the highest AP, followed by much lower scores for "Camping Car," "Pickup," "Truck," and "Van." From figure 9 for text queries, the "Car" category also has the highest AP, but the AP scores for "Camping Car," "Pickup," and "Van" are even lower compared to image queries. The model consistently performs best in detecting the "Car" category across both input types, likely due to the prevalence of cars in dataset. The higher AP for "Car" and "Truck" in both text query types can be attributed to the OWL-ViT model being trained on the COCO dataset using image-text pairs, which includes these common classes. This familiarity results in better performance as the model has been exposed to many examples of pairs of “car” and corresponding images during pre-training. On the other hand, the reliance on text embeddings for classes not seen during pre-training may lead to less effective representations, contributing to lower AP for "Camping Car," "Pickup," and "Van" when compared to image queries. Some sample detections revealing this difference can be found on Appendix B**.** This suggests that image queries might provide more robust contextual information that aids in distinguishing between visually similar categories compared to text queries, which could be limited by how specific or distinct the text descriptions are. Lastly, table 5 shows the number of different type of errors and f1 scores with the best threshold applied to each case (text query and image query). Most common type of error is the classification meaning that the model is able to detect the objects correctly but cannot assign matches with the queries correctly. Therefore, we can conclude that the biggest issue is not in the detector part but in the encoder part which extracts the meaningful features of the images since the matching uses the similarity between those extracted features.

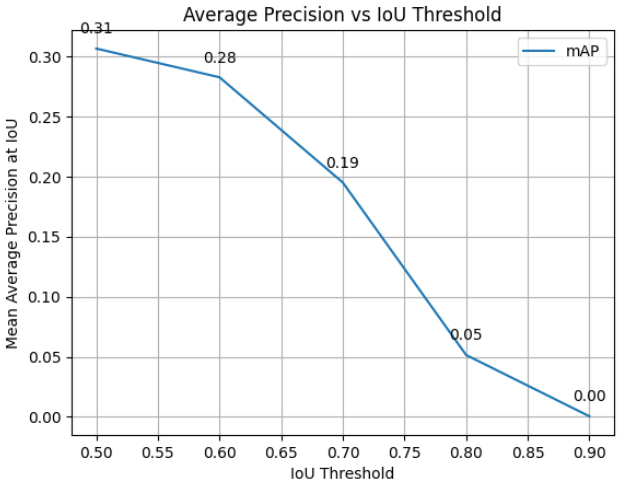

Figure 6: Mean average precision vs IoU threshold of one-shot OWL-ViT with image query

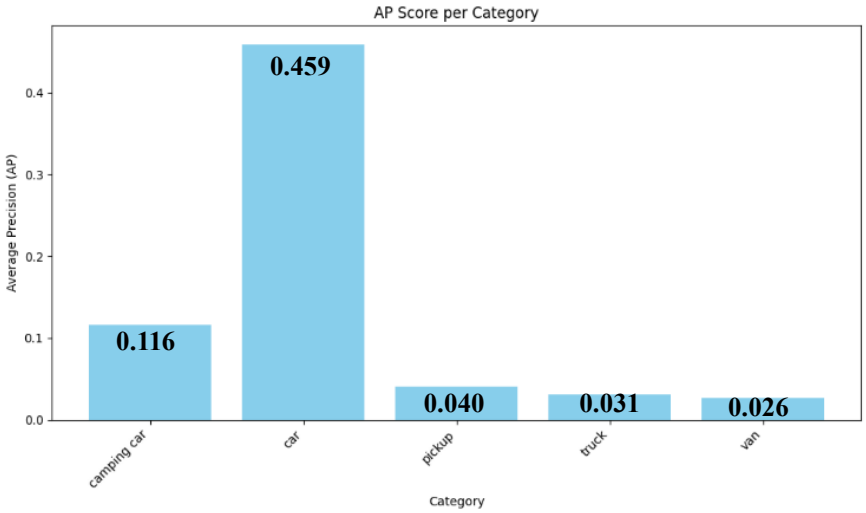

Figure 7: Average precision of each category of one-shot OWL-ViT with image query

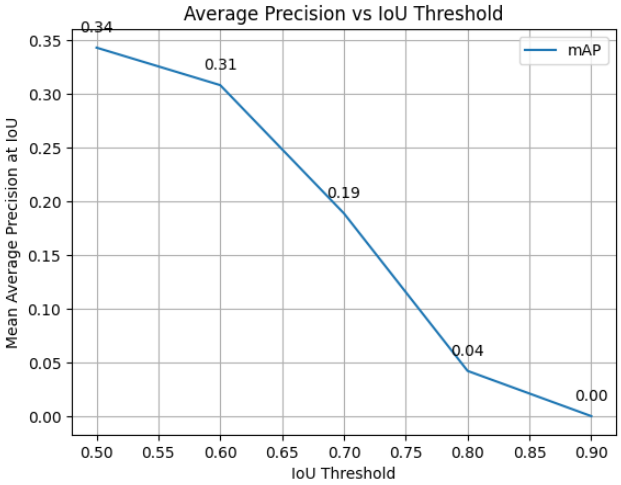

Figure 8: Mean average precision vs IoU threshold of zero-shot OWL-ViT with text queries

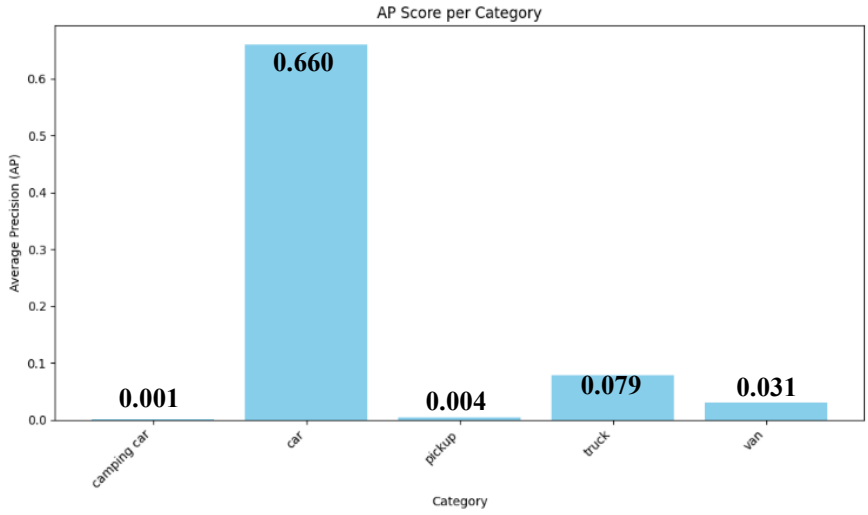

Figure 9: Average precision of each category of zero-shot OWL-ViT with text queries

Query Type	Background	Localization	Classification	Cls & Loc	Duplicate	Correct	Missed	Precision	Recall	F1
Text	149	44	214	17	0	901	617	0.68	0.51	0.58
Image	136	30	386	20	2	740	616	0.56	0.42	0.48

Table 5: F1 score and number of different types of errors comparison for OWL-ViT with zero-shot text and one-shot image queries

5.2 YOLO-WORLD

Table 6 shows the performance of YOLO-World with text embeddings before any training. Most categories show almost zero performance across all metrics (mAP, mAP@50, mAP@75), except for the "car" category, which has a mAP@50 value of 0.204. This result is similar with OWL-ViT since both models are trained on COCO dataset where “car” text query has many corresponding object annotations. Table 7 shows the performance after training with significant improvement in mAP scores of training classes which are "camping car", "pickup", and "truck" with mAP@50 values of 0.716, 0.529 and 0.386 respectively. Since “truck” has the lowest number of samples in the dataset, the performance of that class is considerably low comparing the other two. This training also helps for the model to detect unseen zero-shot classes ("car" and "van") better. However, the mAP values of zero-shot classes remain low even with this improvement. The "car" class increases its mAP@50 to 0.249 while "van" remains under 0.1 in both cases.

Table 8 and 9 shows the performance of YOLO-World with image embeddings before and after training. The model shows zero performance for all classes before training suggesting that the clip image embeddings do not align with the YOLO-World backbone, and it requires fine-tuning. One difference from previous case with text embeddings is that the model is able to show some performance on class “car” even before training because of the pre-trained weights. However, in this case the model has no background on the query image embeddings so it has no clue about the alignment of them with the target images. After training table 9 shows that there is a significant improvement across all classes. Training classes which are "camping car", "pickup", and "truck" reaches mAP@50 values of 0.672, 0.628 and 0.486 showing a more balanced performance between training classes comparing to previous case with text embeddings. Considering the zero-shot performance, class “Car” shows improved performance after training, with a notable increase in mAP@50 (from near-zero to 0.336) This indicates that while the model was not explicitly trained on this class, it has developed a partial understanding, likely due to visual similarities with other vehicles in the training set. However, class “van” has low mAP values even after training (0.063 for mAP@50) and suggesting limited generalization for this class. The overlap in visual characteristics between van and car classes may lead this low performance causing confusion for the model.

To test this, the clip embeddings of query images are plotted with t-sne plot. Figure 10 shows the embeddings of clip query and target image embeddings. This reveals that CLIP vision model has no understanding of the differences between each type of vehicle class and is not capable of clustering the groups of class vehicles together. Therefore, it shows the necessity of fine-tuning CLIP vision model. The t-SNE plot of the embeddings after fine-tuning CLIP is illustrated in figure 11. In this plot, it is possible to differentiate between different types of vehicle classes with correct matches of query and target image embeddings. Finally, table 10 shows the performance when YOLO-World is trained with fine-tuned clustered CLIP embeddings. Comparing to table 9, we can see that there is an improvement in the zero-shot performance. While the map@50 of class “car” increases from 0.336 to 0.356, class “van” shows the most increase from 0.063 to 0.165. Although the overall zero-shot mAP values are still not good enough, this shows the importance of well separated and clustered embeddings on the performance.

category	mAP	mAP_50	mAP_75
camping car	0.0	0.0	0.0
car	0.103	0.204	0.092
pickup	0.0	0.001	0.0
truck	0.001	0.001	0.0
van	0.021	0.041	0.021

Table 6: mAP values of YOLO-WORLD with clip text embeddings before training

category	mAP	mAP_50	mAP_75
camping car	0.493	0.716	0.632
car	0.109	0.249	0.064
pickup	0.33	0.529	0.404
truck	0.297	0.386	0.348
van	0.034	0.061	0.034

Table 7: mAP values of YOLO-WORLD with clip text embeddings after training

category	mAP	mAP_50	mAP_75
camping car	0.001	0.002	0.001
car	0.002	0.003	0.001
pickup	0.001	0.001	0.0
truck	0.0	0.0	0.0
van	0.0	0.001	0.0

Table 8: mAP values of YOLO-WORLD with clip image embeddings before training

category	mAP	mAP_50	mAP_75
camping car	0.457	0.672	0.636
car	0.162	0.336	0.12
pickup	0.417	0.628	0.493
truck	0.356	0.486	0.44
van	0.032	0.063	0.032

Table 9: mAP values of YOLO-WORLD with clip image embeddings after training

category	mAP	mAP_50	mAP_75
camping car	0.575	0.761	0.751
car	0.191	0.356	0.18
pickup	0.285	0.433	0.365
truck	0.425	0.634	0.444
van	0.096	0.164	0.096

Table 10: mAP values of YOLO-WORLD with fine-tuned clip image embeddings after training

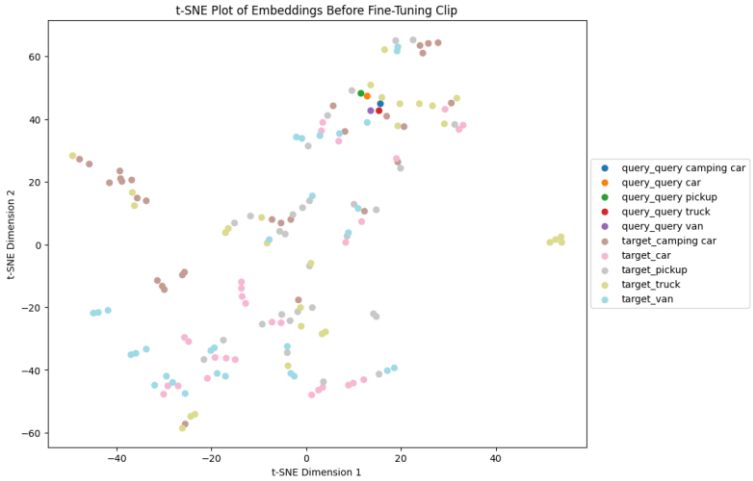

Figure 10: t-SNE plot of the query image and target image embeddings before fine-tuning CLIP

Figure 11: t-SNE plot of the query image and target image embeddings after fine-tuning CLIP

6.0 CONCLUSION

In this project, the performance of two open vocabulary object detection models, OWL-ViT and YOLO-WORLD, are evaluated on vehicle classes in satellite images using both text and image queries. Through detailed analysis of the mAP results across different classes, several key differences are observed in the capabilities and limitations of these models.

OWL-ViT shows a relatively strong performance for zero-shot object detection of class “car”, especially with text queries, due to its pre-training on the COCO dataset using image-text pairs. However, the model struggles to distinguish more specialized classes (e.g., "Camping Car" or "Pickup") with text queries due to the lack of fine-grained vehicle distinctions in the COCO dataset but in the one-shot setting with image queries, it relatively shows better performance for those classes since the model can see the target object during inference and the problem becomes an image similarity problem which is less challenging.

YOLO-WORLD, in contrast, initially performs poorly on zero-shot classes, particularly when only using image embeddings since the model has no information or pre-training about matching those clip image embeddings with the target embeddings extracted by YOLO darknet backbone. After training with image embeddings, YOLO-WORLD demonstrates some ability to detect zero-shot classes, but performance remains limited for visually distinct categories like "van." This indicates that YOLO-WORLD requires fine-tuning and additional training to generalize to new classes effectively. The t-SNE analysis reveals that the model initially struggles to cluster vehicle classes correctly when using CLIP embeddings, but with fine-tuning, it becomes better at differentiating between vehicle types. This reinforces the importance of well-clustered, contextually rich image embeddings for YOLO-WORLD's performance.

6.1 Future Improvements

Fine-tuning OWL-ViT and YOLO-WORLD on a dataset with diverse vehicle types (e.g., different models, and vehicle-specific categories) could improve their zero-shot generalization. Considering that all of the classes except “car” are underrepresented, more training data in these categories might help the models capture unique features of these classes.

YOLO-WORLD benefits from fine-tuned CLIP image embeddings that are better aligned with its object detection backbone. To further enhance performance, more sophisticated embedding clustering techniques, like contrastive learning, could be applied to create well-separated, robust embeddings that capture inter-class differences more effectively.

An ensemble approach combining both image and text queries could leverage the strengths of each modality. For example, using image queries for distinguishing visually similar categories and text queries for high-level object recognition could yield more balanced detection performance.

7.0 REFERENCES

[1] Chargepoly, "Vehicle detection from satellite Dataset," *Roboflow Universe*. Roboflow, May 2023. [Online]. Available: https://universe.roboflow.com/chargepoly/vehicle-detection-from-satellite.

[2] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, 2014, pp. 740–755. [Online]. Available: https://cocodataset.org

[3] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "YOLO-World: Real-Time Open-Vocabulary Object Detection," in *CVPR*, Seattle, WA, USA, 2024. [Online]. Available: https://www.yoloworld.cc

[4] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby, "Simple Open-Vocabulary Object Detection with Vision Transformers," *arXiv preprint arXiv:2205.06230*, 2022. Available: https://arxiv.org/abs/2205.06230.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, X. Weissenborn, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020. Available: https://arxiv.org/abs/2010.11929

[6] J. Jocher et al., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," *IEEE Conference Publication*, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10533619.

[7] W. Wang, M. Liao, Y. Wu, and Y. Li, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," arXiv preprint arXiv:1911.11929, 2019. [Online]. Available: https://arxiv.org/abs/1911.11929

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *arXiv preprint* arXiv:2207.02696, 2022. [Online]. Available: https://arxiv.org/abs/2207.02696

[9] G. Jocher, "SPPF: Spatial Pyramid Pooling Fusion," *GitHub*, Jul. 29, 2022. [Online]. Available: https://github.com/ultralytics/yolov5/issues/8785. [Accessed: Nov. 10, 2024].

[10] A. Radford, J. W. Kim, A. Hallacy, et al., "Learning Transferable Visual Models From Natural Language Supervision," *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, Canada, 2021, pp. 6568-6577, doi: 10.1109/ICCV48922.2021.00646.

APPENDIX A

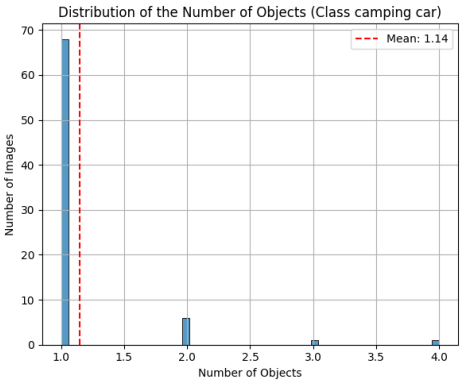

Figure 1: Distribution of number of camping car objects per image

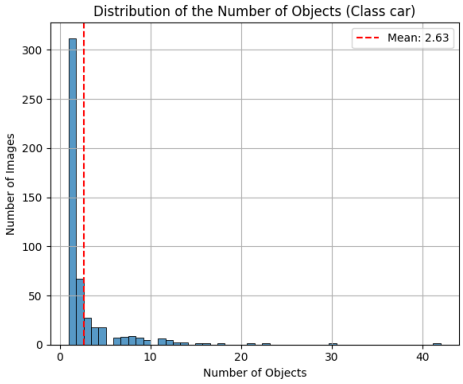

Figure 2: Distribution of number of car objects per image

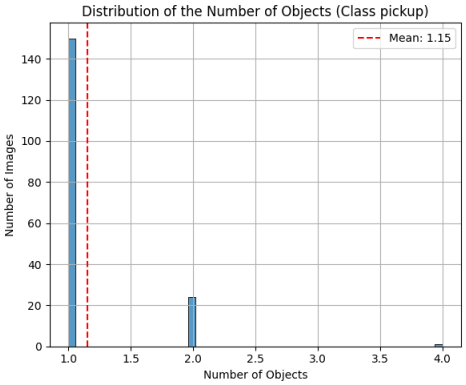

Figure 3: Distribution of number of pickup objects per image

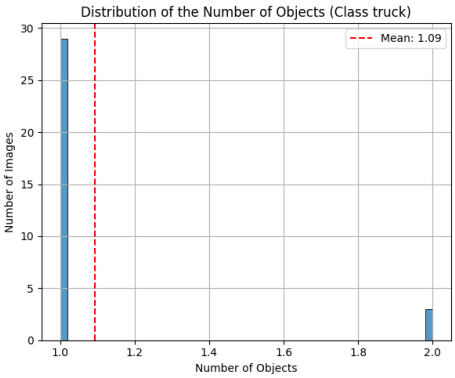

Figure 4: Distribution of number of truck objects per image

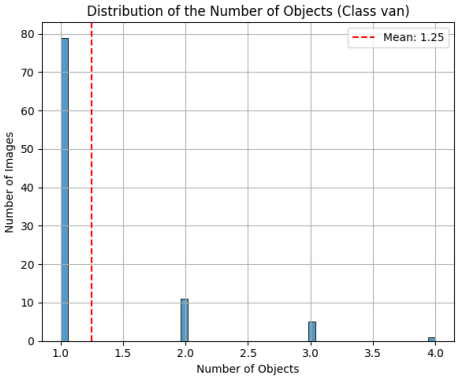

Figure 5: Distribution of number of van objects per image

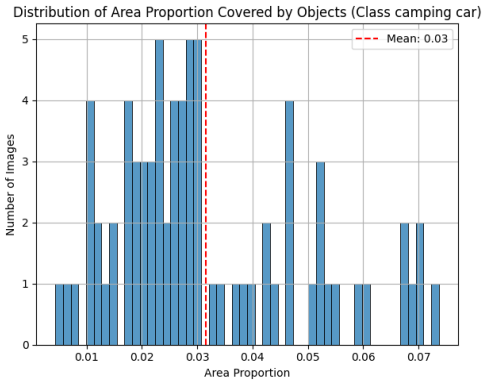

Figure 6: Distribution of area proportion covered by camping car objects per image

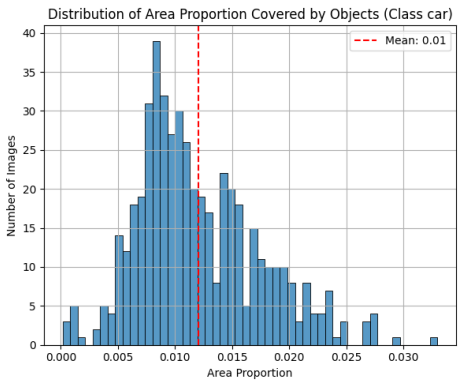

Figure 7: Distribution area proportion covered by car objects per image

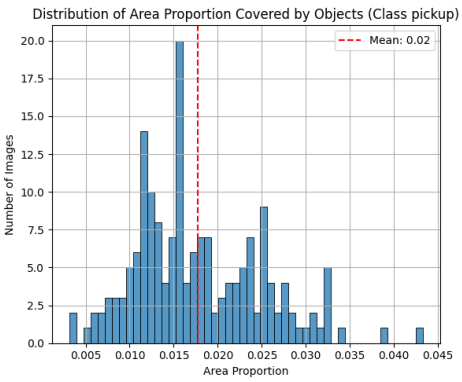

Figure 8: Distribution of area proportion covered by pickup objects per image

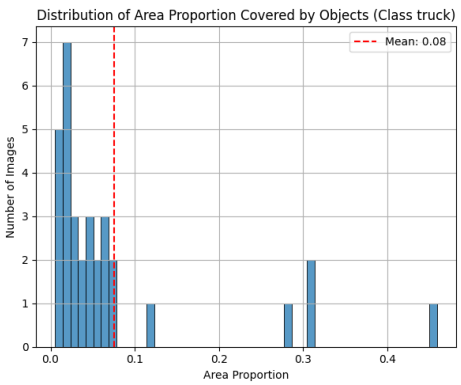

Figure 9: Distribution of area proportion covered by truck objects per image

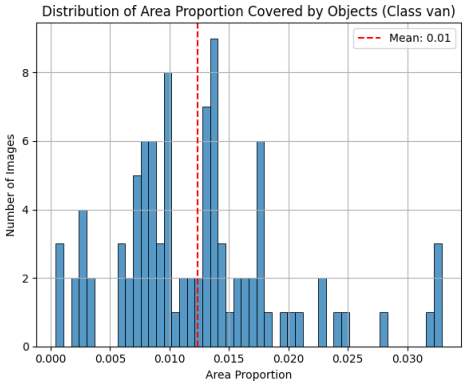

Figure 10: Distribution of area proportion covered by van objects per image

APPENDIX B

Figure 1: Sample detections of OWL-ViT with image query

Figure 2: Sample detections of OWL-ViT with text query

APPENDIX C

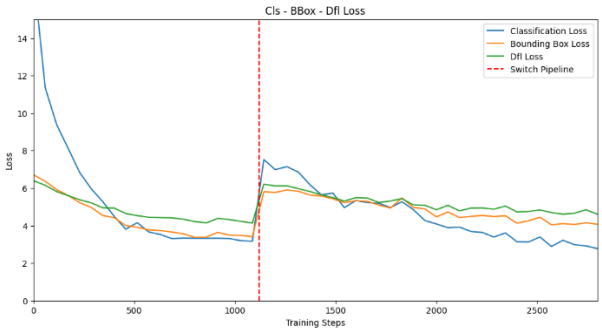

Figure 1: Classification, bounding box and Dfl loss history during training of YOLO-World with text embeddings

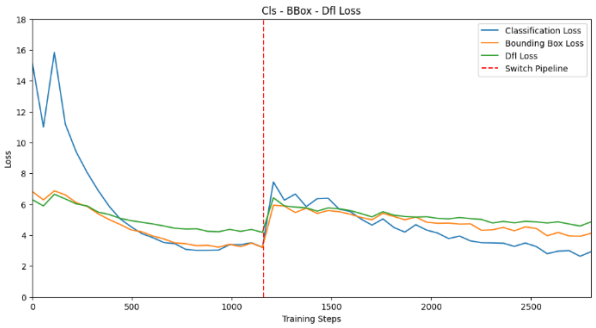

Figure 2: Classification, bounding box and Dfl loss history during training of YOLO-World with original CLIP image embeddings

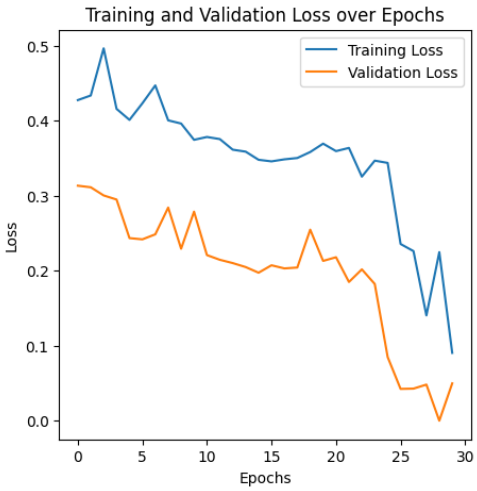

Figure 3: Training and validation loss history of CLIP vision model during fine-tuning of image embeddings for YOLO-World

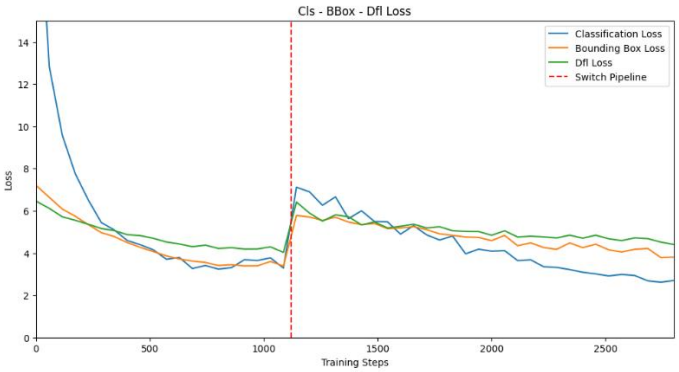

Figure 4: Classification, bounding box and Dfl loss history during training of YOLO-World with fine-tuned CLIP image embeddings

APPENDIX D

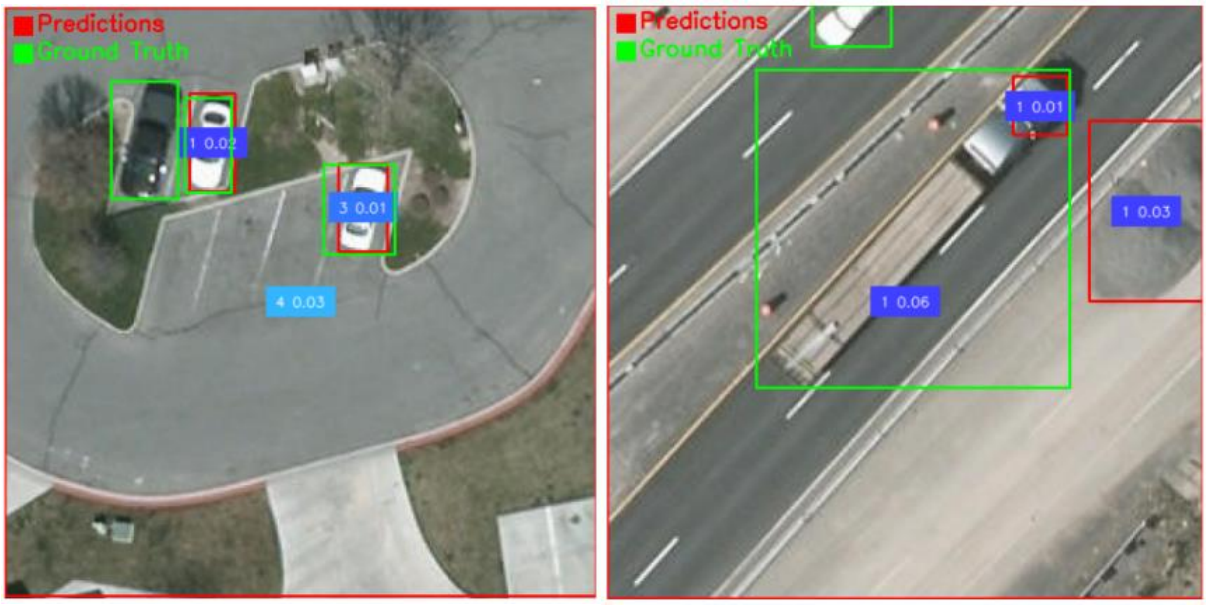

Figure 1: Sample detections of YOLO-World with text query before training

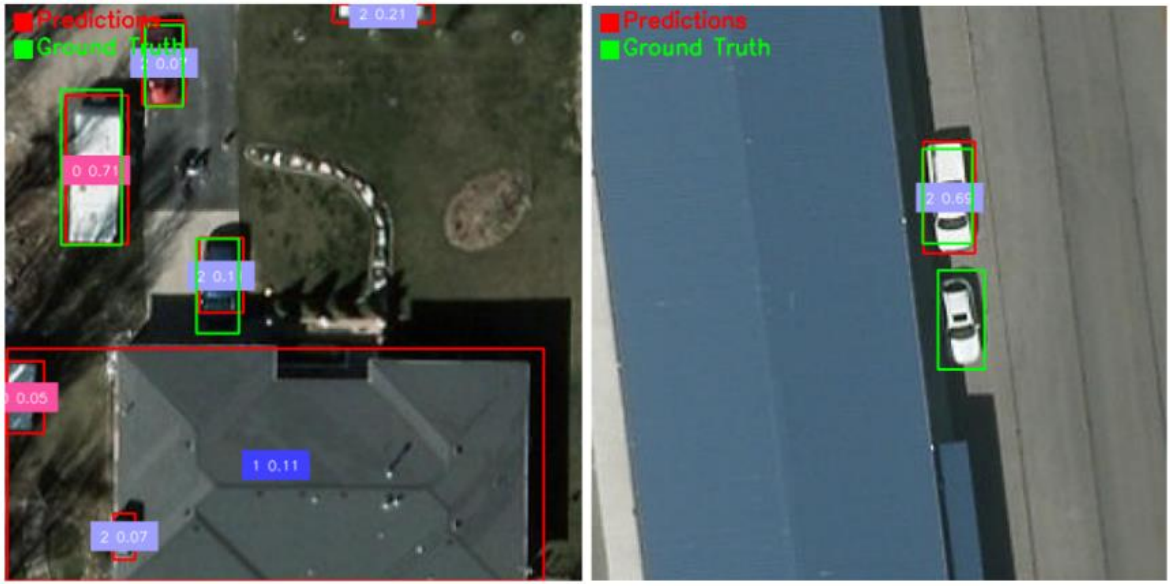

Figure 2: Sample detections of YOLO-World with text query after training

Figure 3: Sample detections of YOLO-World with image query before training

Figure 4: Sample detections of YOLO-World with image query after training

Figure 5: Sample detections of YOLO-World with fine-tuned image query after training