INTEGRATING VLMs IN OBJECT DETECTION

SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

24/25-CMT405 NLP DISSERTATION

AUTHOR: IAN O. RODRÍGUEZ ALMODÓVAR

SUPERVISOR: DR. JING WU

MODERATOR: USASHI CHATTERJEE

TABLE OF CONTENTS

1 - Introduction

The rapidly evolving field of Artificial Intelligence is a tree that has grown numerous branches, often in vastly different directions. At some point down the line, the field of Natural Language Processing resulted in the creation of robust multimodal Large Language Models, which are useful for not only the vast majority of text-based tasks such as research, editing, and generation, but also the ability to ingest, understand, and process data in all of its different formats, not the least of which is images.

The task which will be worked on in this paper is known as **Remote Sensing**. As defined by the United States Geological Survey, remote sensing is about detecting and monitoring the physical characteristics of objects and areas by reading their emitted radiation from a distance, whether it be from a satellite or from an aircraft. This idea is also true in Remote Sensing tasks related to machine learning, as a subcategory **of Object Detection**. Object detection tasks have been mostly limited to tasks relating to self-driving vehicles, and detections of small objects in non-remote sensing scenarios. Remote sensing tasks are notoriously difficult in comparison, most notably due to the fact that most available VLMs, MM-LLMs or even Open Vocabulary Object Detectors are not necessarily trained on detecting objects (and more so small objects) from above, most of which are only a few pixels wide. This makes the task dependent ultimately on image resolution, which in turn requires more robust and expensive equipment to perform experimentation on. The need for more inexpensive approaches is thus a necessary area of research, and why it is the central purpose of this research project.

At a larger scale, remote sensing tasks are extremely important for environmental purposes as well. Namely, it helps us understand the state, conditions and properties of the regions. Whether it's to investigate the state of crops and predict yield, training machine learning models to recognize the conditions that precede natural disasters, or the health of ecosystems; there are innumerable benefits to research and development in efficient methods for remote sensing. A study by Akyar Akyar et al. (2024) named "Deep artificial intelligence applications for natural disaster management systems: A methodological review", provides a comprehensive description of AI methods used for natural disaster management in three categories: earthquake, flood, and forest fire disasters. The most popular network structure used for these tasks was the Convolutional Neural Network (CNN), among others. They concluded that semantic segmentation allows for more abundant and complex spatial understanding, and CNN architecture, when used on high resolution satellite image, shows very good performance distinguishing the background from the active fires. There was also a conviction that remote sensing technologies are essential for managing natural disasters in all its stages; before, during, and after. This makes the impact and motivation of research done in this field crystal clear.

The pipeline this project proposes is rather simple: we use a teacher Yolov5 model, which we train and validate on the xView dataset, to produce labels. These labels are then passed through two methods of enhancement which are complementary but implemented separately in order to assess their effectiveness in improving mean average precision, precision and recall in their respective Yolov5 student models.

- **Owlv2** by Google is used for refinement of teacher bounding boxes.
- **RS-CLIP** is a CLIP model fine-tuned on remote sensing datasets which is used for the purpose of filtering out false positives.
- Combining **both** approaches to produce enhanced labels.

After all of the three separate artifacts have been exported, three fresh Yolov5 students are trained on the enhanced labels. Each Yolov5 student is trained for 25 epochs, enough for the Yolov5 model to generalize on the enhanced labels to then be run on unseen data to observe their performance. Performance on the test data is then compared to the rest of the student models, which should show us which of these three steps help generalization the most.

This research project is based on a modality which is curiously named "Teacher-Student Framework". It is a pipeline in which we train a **teacher** detector, run it on a training set, and export the labels. These labels we now call pseudo-labels, those created by a teacher model. The **student** is the model that is trained on these generated pseudo-labels, generally with the hope that with more data, it will achieve better results. This project is centered around using an OVOD (OWLv2) and a VLM (RS-CLIP) to process these pseudo-labels by using semantic or linguistic information, also with the hope of allowing these student YOLOv5 models to generalize better on a complicated and imbalanced dataset, commonly known as xView.

2 – Background/Literature Review

2.1 Vision-Language Models

The task of object detection is a notoriously difficult one, and as a result, computer vision scientists have increasingly been turning to the field of NLP looking for answers, believing possibly that grounding objects and areas in linguistic information might be a way to improve or refine the capabilities of object detection models when it comes to identifying and classifying objects. In "A Survey on Vision-Language Models and Object Detection" by Yiliang Chen et al. (2025), it is stipulated that the first vision-language models consisted in using CNNs and RNNs to encode images and text into separate feature vectors and then simply fusing them; other approaches would employ a CNN to encode the features and an RNN would generate captions for those feature vectors.

The introduction of transformer-based models and their self-attention mechanism brought with it attempts at using this new innovative technology to create pipelines combining these LLMs with vision models. For instance, VisualBERT by Liunian Harold Li et al. (2019) created a framework in which they combined BERT with object proposal systems such as Faster-RCNN. This bridge between vision and language proposed by VisualBERT yielded promising results, often outperforming contemporary models while being quite simple to implement. Another well-known example of a vision-language transformer-based model is CLIP (Contrastive Language-Image Pre-training), developed by OpenAI in 2021, which is based on the utilization of a pair of neural networks: one of which is focused on understanding images, and the other for understanding text. CLIP was originally introduced in a paper by Alec Radford, Jong Wook Kim et al. (2021), and its performance on visual classification benchmarks is impressive to say the least. What is so innovative and efficient about CLIP is the fact that simple pre-training is enough to have it perform at highly competitive levels. CLIP's objective is contrastive, which means that it has an embedding space inside which it attempts to bring the most ideal image-language pair while also distancing them from all other mismatched pairs. This is what allows CLIP to do zero-shot classification while also yielding competitive performance. It also allows testing and utilization in low resource settings, which helped in this experimentation. State-of-the-art multimodal LLMs are also effective for zero-shot training and performance, however these are notably heavier and may be more expensive in terms of resources than CLIP.

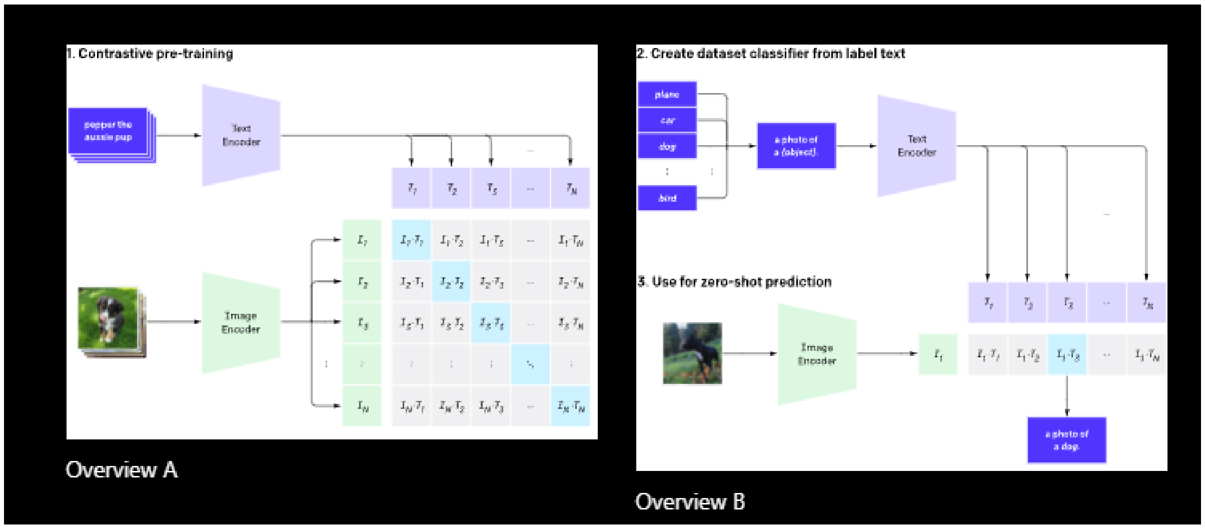


FIGURE 1: CLIP ARCHITECTURE

2.2 Deep Learning Object Detection

The experimentation which was conducted and is presented in this paper relies specifically on the most well-known and possibly best-performing deep learning object detection models: the YOLO models. This family of deep learning object detection models were introduced by Joseph Redmon et al. (2015) in their paper "You Only Look Once: Unified, Real-Time Object Detection". At its core, the YOLO model is simple: it first resizes the image its given, it then runs a convolutional network on the images and then sorts the resulting detections by the model's confidence. This metric is a very important feature that is used in this paper by external agents like VLMs to verify or exclude object detections. Looking at it more closely, YOLO will run this convolutional network to produce bounding boxes over the objects it detects on the image. In the authors' own words: "we reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates". The model will tile the image into a grid, and for each grid cell it produces bounding boxes, the coordinates of which consist of 5 predictions: the notable x, y coordinates which represent the center of the box relative to the grid cell in which it is enclosed; and width and height, which are produced relative to the image at large.

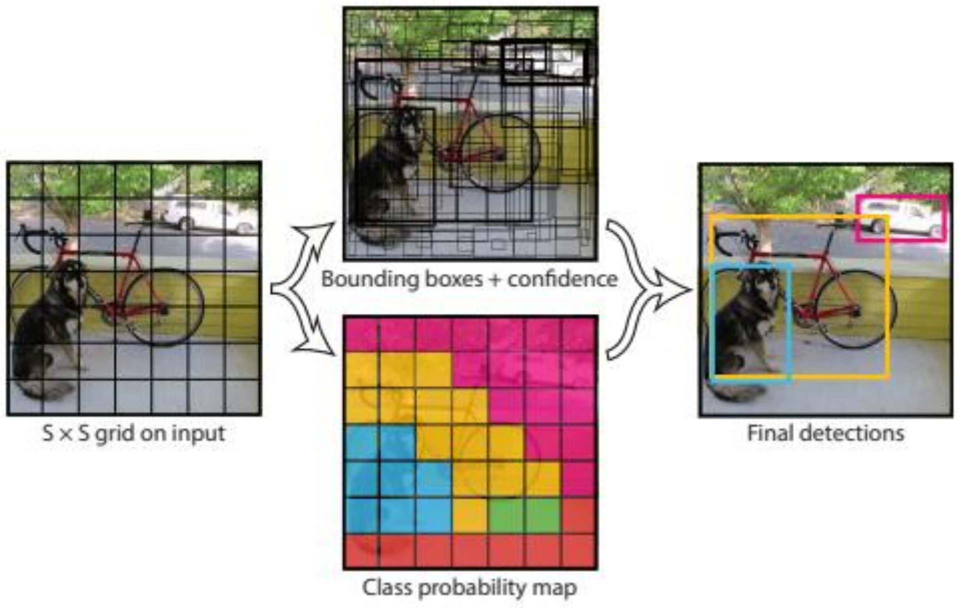


FIGURE 2: YOLO PIPELINE

A comprehensive study of the evolution of the YOLO family of models by Nidhal Jegham, Chang Young Koh et al. (2025), demonstrates the performance of most previous versions of YOLO by examining their performance in three datasets: Traffic Signs, African Wildlife, and Ships and Vessels. They note that one of the biggest obstacles when training YOLO models is the disparity between Mean Average Precision 50-95 and Mean Average

Precision 50; this means that there is a particular challenge with higher IoU thresholds in the detection of **smaller objects**.

Further observation of the discrepancies between the three different datasets they used, the YOLO models achieved very similar and competitive performance in the Traffic Signs and African Wildlife datasets. The Traffic Signs dataset is notably simpler, since it contains images of traffic signs in diverse settings, but still ultimately traffic signs. The Africa Wildlife Dataset is also notably simple, containing only four common African animal classes: buffalo, elephant, rhino, and zebra, all in very diverse natural settings. However, performance dropped considerably when measuring on the Ships and Vessels dataset. It contains only one class called "ship" and is notably massive: containing 13.5k images. When it came to the overall performance of the models, across their generations, performance did vary but unremarkably so.

2.3 Open Vocabulary Object Detection

Chaoyang Zhu, Long Chen et al. (2024) have published a very extensive and robust survey on the development of this fascinating area of object detection and NLP. The concept of open vocabulary learning and its development was sped up after the introduction of CLIP which allows this to happen within its embedding space, where it takes a contrastive approach by comparing correct image-language pairs with incorrect ones, pushing the incorrect ones further away. OVOD models are excellent for low resource settings for this reason; you can map a text prompt containing a class of object and compare it to the object in the image after being both encoded into embeddings. The need for massive, annotated datasets is thus reduced. Chaoyang Zhu and Long Chen in their paper said it best when they said that zero shot detections were proposed to "allow the access to any unannotated, unseen visual object". This paper brought to light the very interesting concept of **Pseudo-Labeling**, which we work with in the experiments presented on this paper.

The introduction of **Grounding DINO** was a significant leap forward in the field of OVOD. First introduced by Shilong Liu, Zhaoyang Zeng et al. (2024) in a paper called

“Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection”, it is a fusion between DINO, which is a transformer-based detector with grounded pre-training. As stated in the introduction paper, the purpose of Grounding DINO is to provide “a strong system to detect arbitrary objects specified by human language inputs”, this is in relation to their overarching goal of contributing to the development of an AGI (Artificial General Intelligence). Grounding DINO is emblematic of the work being done in OVOD, as it consists of utilizing contrastive loss, much like CLIP. Its pipeline is more complex; it contains a text and an image backbone which are turned into features and enhanced through a feature enhancer. The enhanced text features are then kept and used for Image Cross-Attention and Text Cross-Attention in the decoder layer for the enhanced image features. The model outputs are then used to measure contrastive loss and localization loss.

The model used in these experiments, however, is called **OWLv2**. This is a model introduced by Google Research in their paper by Matthias Minderer et al. (2024). Its pipeline is notably simpler than Grounding DINO’s, containing only 3 steps (one of them being optional). It generates pseudo-box annotations, trains new models on those pseudo annotations, and one can then optionally fine-tune on human annotated datasets. Generation of these pseudo annotations are made from the WebLI dataset by using OWL-ViT CLIP L/14, which contains 10 billion image-text pairs, which are then passed on for self-training by the other OWL-ViT models. The authors noted an unignorable limitation to this model, which is the fine-tuning and open-world performance trade-off. Fine-tuning this model will aid in generalization of that specific dataset, but it will be at the cost of the model’s open-world performance.

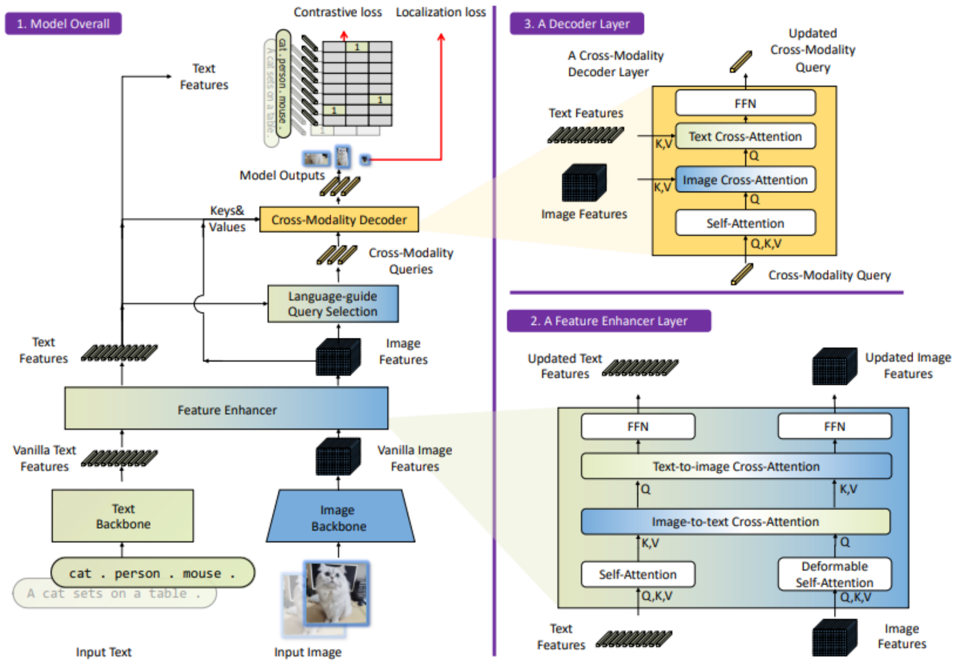


FIGURE 3: GROUNDING DINO ARCHITECTURE

2.4 VLMs and OVODs for refinement and verification

A study by Hong Lu et al. (2025) introduces an approach they named ClipGrader, it is an attempt to use CLIP to evaluate in detail the bounding box and the accuracy of the object label. This interesting approach has significant challenges, mainly that it requires CLIP to detect objects within bounding boxes, which requires an understanding of the context surrounding the object and is very dependent on the quality of the image itself. They are the inspiration for using RS-CLIP for label verification and FP filtration. They take CLIP and run it on a bounding box drawn on a magenta color to avoid confusion with the background. They then give CLIP a set of class-aware prompts for it to determine whether it can determine if the class associated with the object inside of that bounding box is a good one, a bad one, or just background. Furthermore, while not directly about box refinement, open vocabulary object detectors have been used in order to decrease dependency in human annotated object detection datasets by increasing categories of objects through novel zero-shot detections. For instance, a paper by Mingfei Gao, Chen Xing et al. (2022) approached this problem by using a pre-trained VLM that generated bounding box labels for objects and then used those pseudo-labels to improve open vocabulary detection. This resulted in their method beating their strongest baseline (citing another study by Zareian et al.) by about 8%. It also consistently beats it in datasets outside of COCO, the dataset they fine-tuned on. This study also calls for more research in utilizing pseudo-labels to not only train OVODs but also stronger object detectors such as the YOLO models.

2.5 The Teacher-Student Framework

The framework of pseudo-labeling as we know it today can be linked back to a study by Dong-Hyun Lee in 2013, named "Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". In this paper, pseudo-labeling is defined as a semi-supervised training framework in which a machine learning model is trained on a dataset which has been infused with unlabeled as well as labeled data. The point of this framework is to improve the performance and generalization of machine learning models by using unlabeled data, and by using the model's own prediction on any unlabeled image as a ground truth, while also doing regular supervised training on human annotated or labeled data. This is different to what is done in this research paper, in the sense that we are only doing self-training with pseudo-labels, where the same are curated. Qizhe Xie et al. (2020) take a similar approach to Dong-Hyun Lee in the sense that they also propose a semi-supervised learning approach in which they use unlabeled data to improve performance on SOTA ImageNet accuracy. They run a teacher model (which is already fine-tuned on labeled data) in order to generate pseudo-labels from unlabeled, which are then treated as ground truth labels for the student model to train on. The student model is thus trained on a dataset which has a combination of pseudo-labels and ground truth labels. This is similar to what we do in this paper in the sense that we augment the pseudo-labeled images with ground truth labels in order to account for classes that may have been missed by the teacher model.

3 – Methodology

3.1 The Dataset

The xView dataset was released in 2018 by the National Geospatial-Intelligence Agency in collaboration with the Defense Innovation Unit Experimental. It is one of the largest publicly available remote sensing datasets which contain satellite imagery and 60 categories of objects. Many of these categories are small cars, different kinds of trucks, and a diverse variety of small objects which make detection quite difficult and resolution dependent. Furthermore, the images are very high in resolution, around 30 centimeters per pixel. This helps make small objects such as cars appear clearer for detection. However, training on these entire images is computationally expensive to say the least. It was therefore necessary to tile the dataset into 1536x1536 pixels as well as implement a 386-pixel stride.

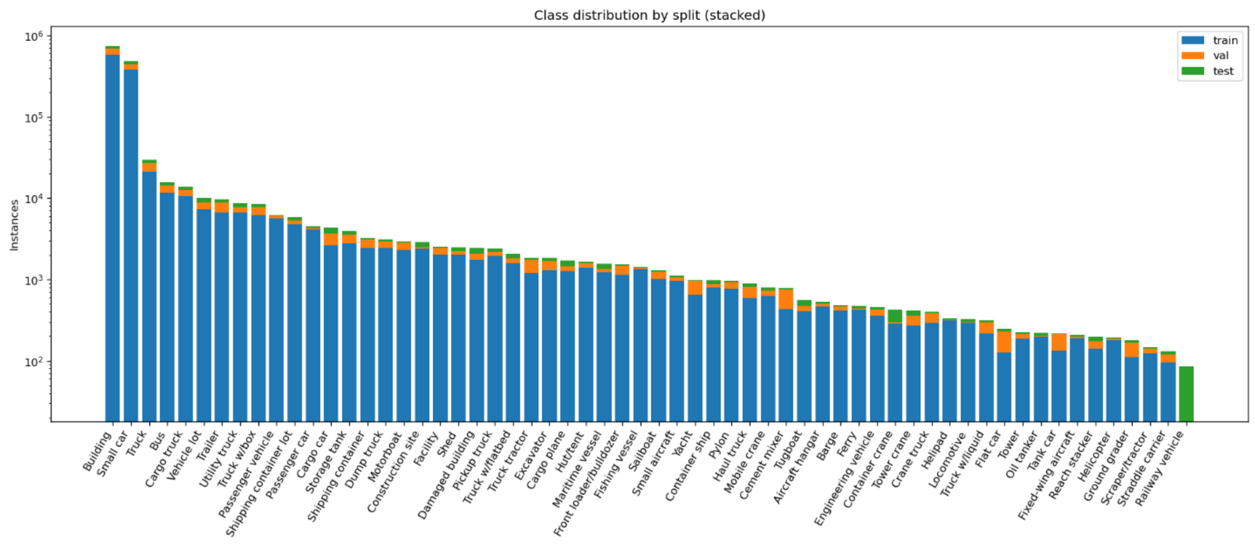
Training and experimentation were only done on an RTX 4070 laptop with 8 GB in VRAM, so resources were very limited. The choice of image resolution and stride is to maintain a fair quality of small objects for latter cropping, for detection, and for later label enhancement via the VLM and OVODs. Stride at 386 pixels and a resolution of 1536x1536 means that we're overlapping 75% in each direction. This is done with the intention of keeping border or edge objects during image tiling. This makes training the Yolov5 models quicker and is vastly less computationally expensive.

3.1.1 Challenges

It is also extremely unbalanced: it contains classes such as Buildings and Small Cars that contain tens of thousands of instances, but also classes that have a couple hundred instances and sometimes even less than that. This causes YOLOv5 to get biased toward buildings and falsely label structures such as storage tanks, facilities and aircraft hangers as buildings; at times it will also classify as buildings empty patches of grass or trees. Classes that require other objects such as vehicle lots and construction sites are also difficult for the YOLOv5 model to generalize on, less so for vehicle lots, but the model sometimes falsely detects construction site objects in empty patches of desert. Furthermore, construction vehicles such as excavators are sparser and their presence is virtually invisible when compared to small cars and trucks.

The aforementioned weaknesses and imbalanced nature of the dataset may be the cause of decreased performance levels in precision, recall and mAP, which are the most important metrics we will use.

FIGURE 4: CLASS DISTRIBUTION BY SPLIT



The imbalanced nature of the dataset causes also an inadvertent effect: what happens to the classes that don't get detected in the baseline YOLOv5 model? This poses a problem for the YOLOv5 students, because the baseline model will fail to classify objects of which there are only a few hundred instances (or boxes). And if the YOLOv5 baseline missed them, its students will too if they train only on their pseudo labels. This last scenario was tested and the YOLOv5 baseline performed even worse than its teacher. This is why we performed an extra step, a sort of **backfill**. This backfill added 25,776 GT boxes of classes that were not represented in the pseudo-labels, which were significant (only about 35 of 60 classes were represented). This is in line with our main goal: to find ways to increase performance of object detectors using refinement and filtering by utilizing semantic information. Moreover, allowing the training data to be majority pseudo-labels is also in-line with the goal of experimenting with pseudo-labels to observe performance.

This is not merely an option but a necessity: as the xView dataset is much too imbalanced, and classes that are missed result in drastically lower mAP and overall recall.

3.2 Dataset Preprocessing

The xView dataset doesn't come ready to be passed into a YOLOv5 model, it has to be preprocessed first into a format that it can understand. First, we do deterministic splitting by seed, so we map each file to a random value between 0 and 1. We're treating the basic dataset as, essentially, a pool from where we assign whether a file goes into the train, validation, or test split. Since this research project consists of comparing and contrasting the performance of different object detection models, it is beneficial to have clean, controlled splits. The split is standard: 0.75 for train, 0.15 for validation, 0.10 for test.

The images came in a TIF format, which are very heavy; therefore, it was necessary to convert the images to JPEG. This step may result in some loss of image quality, but for the purposes of this research, it has been proven to be rather negligible. Converting from TIF to JPEG also reduces the space taken up by the images on the disk (this is necessary as we're working in a rather low resource setting).

Another important step that needs correcting is how xView annotates its images. As it was observed earlier, YOLO models expect a certain notation to recognize where a bounding box is drawn. xView annotations are in a x1, y1, x2, y2 format, these represent pixel coordinates. However, Yolo necessitates normalized center, width and height values: x, y representing the center of the box relative to the grid, the width and height of the bounding box itself. A function was thus built to perform this conversion.

YOLO models also expect a file in yaml which points to the directories where the training images and labels are found. While it is not necessary to have class names for training, in order to experiment with VLMs it was required to make a class list in Json format that mapped the class identifiers to the class names. This is also needed for exploratory data analysis and visualization when exporting overlays for auditing purposes, such as to check what classes the YOLO model is having the most trouble detecting.

3.3 RS-CLIP

RS-CLIP likely comes from a paper by Xiang Li, Congcong Wen (2023) in a paper titled "RS-CLIP: Zero-shot remote sensing scene classification via contrastive vision-language supervision". In this paper, RS-CLIP was noted to have competitive performance in zero-shot and few-shot learning classification in remote sensing. What is done in this experimentation is using OpenCLIP with remote sensing prompts such as:

"a satellite image of a {class_name}"

"An aerial photo of a {class_name}"

"An overhead view of a {class_name}"

"a top-down remote sensing image of a {class_name}"

"a high-altitude view of a {class_name}"

These prompts nudge OpenCLIP towards recognizing the objects it's given as being observed from an overhead perspective. CLIP is used here as a semantic auditor or referee: we use it to keep or drop bounding boxes, as well as recheck objects within bounding boxes with context. This is done by cropping the bounding boxes from the preprocessed tiles and encoding them with CLIP, we then also encode the class identifiers with the aforementioned overhead-tuned templates; they are averaged to one vector per class.

After computing logits, we tune the parameters to nudge CLIP to be conservative and careful with the boxes it drops, therefore having it lean towards keeping the boxes. YOLOv5 is already relatively good at detecting most categories of objects, so dropping a lot of boxes may hurt performance by reducing true positives, killing recall and hurting mAP.

The parameters for CLIP are straightforward: if teacher label confidence is more than 0.40, we consider it a high confidence for an object and keep it automatically. If the pixels of a small object are less than 6, we keep that object. This is due to the fact that small objects are so vanishingly small that CLIP cannot be trusted to reliably detect and associate with a class name. For cases in which CLIP may be unsure about dropping, we allow it to do a context pass, effectively allowing it to look at the rest of the tile and re-encode. It then combines the crop with the context and performs the decision once more.

These parameters were decided with balance as an ideal: it is necessary for there to be enough drops to warrant some kind of visible effect on the YOLOv5 student model we are training on its enhanced labels. Moreover, we turned CLIP's similarities into probabilities in order to create a risk score, which would compare the target probability versus the best alternative. Each box is classified as either keep (if the teacher model has high confidence in it), red (if there's a clear mismatch), or amber (borderline, the model is insecure). The red boxes are always dropped, and amber boxes are only dropped relative to the amount of the boxes we have dropped globally; we aim for 50,000 dropped boxes at least so that we see some visible effect on YOLOv5 student model performance later, and with the hope that these boxes will just be background, which we know for sure that the YOLOv5 model has some problems with as it tends to detect objects often where it's clear that there are none.

Once filtering is done, we extract the enhanced labels and keep a CSV file containing information about the filtering that was done for auditing purposes.

3.4 OWLv2

OWLv2 is an Open-Vocabulary Object Detection model developed by Google DeepMind in May of 2024 by Matthias Minderer, Alexey Gritsenko et al. This model is purported to have improved AP by a considerable amount (from 31% to 43%) for rare classes in the LVIS dataset. In its introduction paper, OWLv2 is not used on remote sensing datasets, so it is a point of interest of this paper to see its performance in this kind of setting, with objects that are small and hard to detect or have a structure that is hard to recognize.

While OWLv2 has shown been shown to improve AP in rare classes for which it has not necessarily seen human box annotations, it is not used for this purpose here. The central purpose we have for OWLv2 for these experiments is to simply **verify** or **refine** the teacher YOLOv5's bounding boxes, for the purposes of maintaining recall at a high level and possibly also increasing mAP as the two priorities.

To do this, we follow this step-by-step procedure: for all the teacher's bounding boxes we get its corresponding class identifiers, then we convert each class name into a prompt with prompt templates like the ones we used for CLIP. We cap the max text labels at 80 per image and then deduplicate. We then run OWLv2 once per image; a forward pass containing the image along with the list of prompts. Once this is done, we process the detections to return bounding boxes in pixel coordinates and also return confidence scores, and text labels.

Each detection made by OWLv2 is then grouped by text labels, such that we only end up comparing consistent objects (a small car to a small car, a storage tank to a storage tank, etc.). The teacher's labels are then converted from YOLO coordinates to pixel coordinates. Afterwards, among the OWLv2 boxes we grouped for the same prompt, we find the best IoU match. For a detection to be verified, the confidence score for OWLv2 has to exceed 0.15, and the IoU has to be 0.50 at a minimum (as long as the overlap between the OWLv2 box and the target box is more than 50%, it passes the gate). If the gates are passed, the detection is verified. After a detection has been confirmed, it passes onto the refinement stage.

The refinement stage consists of essentially snapping the teacher's bounding box to OWLv2's bounding box, which may be better aligned to the object than the former. This only happens when OWLv2 is confident about its own bounding boxes. If there is no qualifying match, we keep the box as it is. The reason we do not drop bounding boxes in this step is to avoid killing mAP, precision, and recall, which are already low according to the YOLOv5 baseline model results on the training data. These will be presented later.

4 – Metrics

In order to evaluate how the models are performing, we employ the use of the common metrics used across the machine learning world: precision, recall, and F1-score. However, we also make use of some new metrics such as mAP@0.5 and mAP@0.5:0.95, and IoU. These last three are a common sight in the world of object detection and especially in remote sensing experimentation.

The metric mAP@0.5 is the average precision over all classes in a dataset when a prediction counts as correct. A prediction is only marked as correct if its bounding box overlaps with the GT (ground truth) box by more than a 0.5 IoU. IoU here stands for **Intersection over Union** and is computed as the overlap area divided by the union area between the predicted and the GT boxes. Intuitively, mAP@0.5 can be defined as a forgiving metric that measures whether the object was found *somewhere* in the right place. However, mAP@0.5:0.95 is more precise. It checks whether the object was found and whether it was precisely localized.

5 – YOLOv5 Training Phase

5.1 The Baseline

Interestingly, when YOLOv5 was trained for this experimentation on the preprocessed and tiled xView dataset, precision scores are about as high as in studies such as the one by Matthew Ciolino et al. (2023); but recall, mAP@0.5, and mAP@0.5:0.95 remained extremely low. This is even after training for 50 epochs.

As can be seen in Figure 5, as precision decreased, all other measures increased very slowly but steadily across epochs. The YOLOv5 weights were downloaded and used locally; this particular model was pre-trained on the COCO dataset. The same image size with which we preprocessed and tiled the dataset (1536x1536) was also set during training, in order to allow the model to have good enough resolution to detect small objects that may be otherwise difficult to detect at lower resolutions. When we look at Figure 6, it is more than clear that the model is extremely conservative and barely makes predictions.

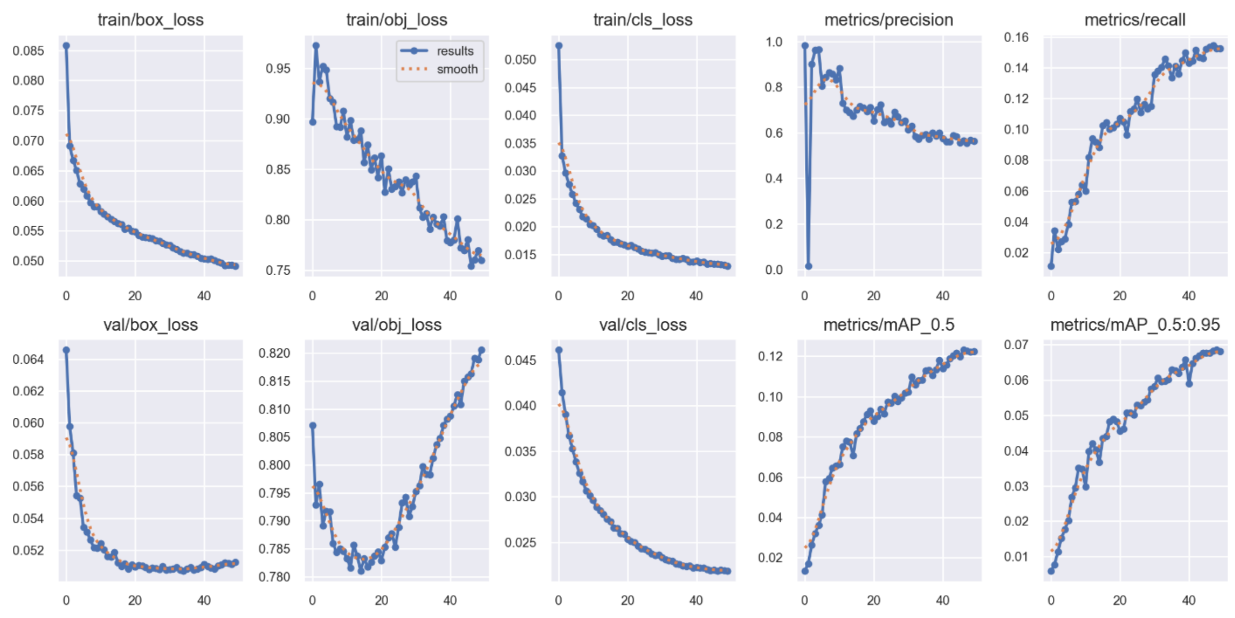


FIGURE 5: YOLOv5 BASELINE RESULTS

The bright line at the bottom of the confusion matrix on Figure 6 shows how often the model will overpredict objects as just background, which means a quite a significant number of False Positives. This could mean a number of different things, like many detected objects falling below the confidence threshold. It also seems to lack the context required to make distinctions between different kinds of objects that belong to a same class. For instance, in order to for the model to be able to differentiate between a vehicle lot and a construction site, it has to recognize whether there's indicators like excavators, bulldozers, or cement mixers or alternatively, small cars, flat cars, or trucks. It is also clear that vehicle lots are more likely to be True Positives than construction sites, possibly due to the fact that the Small Car category has the second greatest number of instances in the entire dataset, only after buildings. The model may also overpredict buildings for any upright structure, so structures such as facilities and storage tanks are also in danger of this.

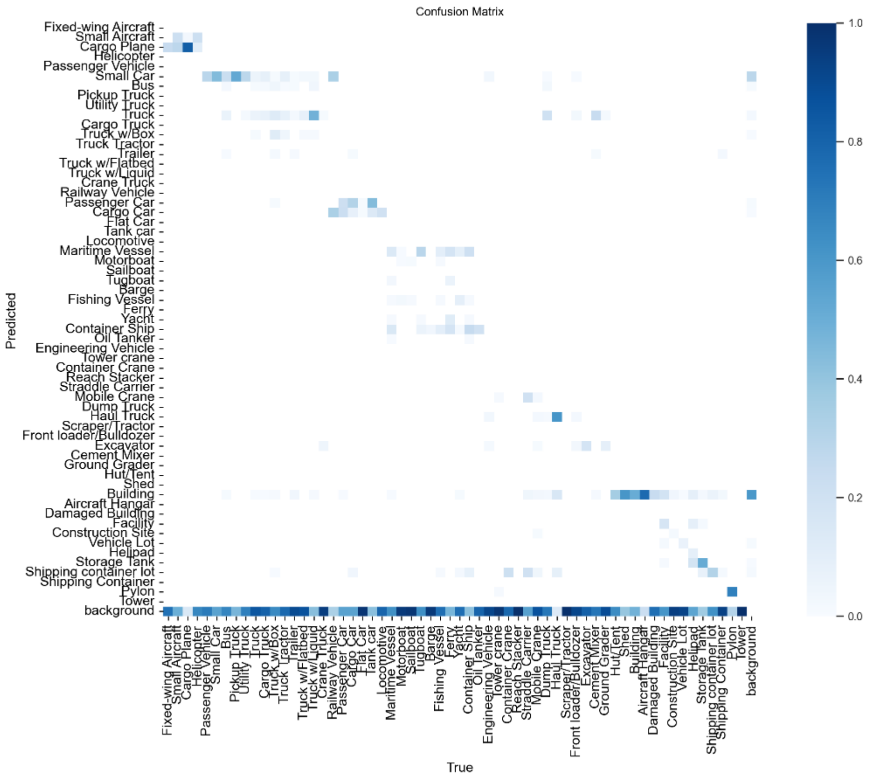
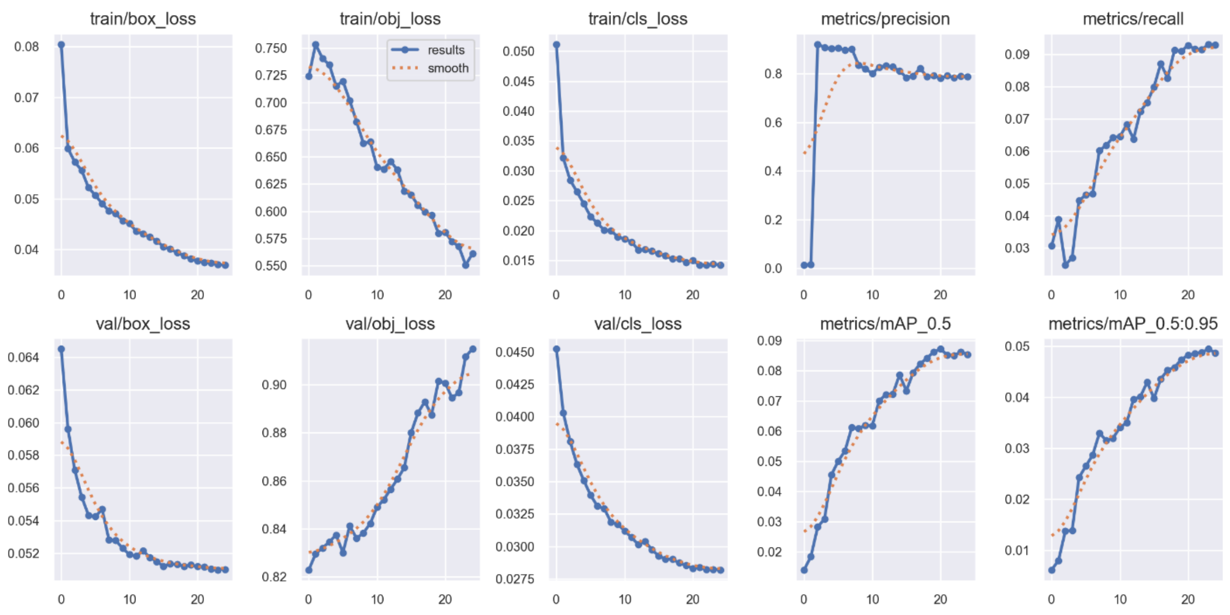


FIGURE 6: YOLOv5 BASELINE MODEL CONFUSION MATRIX

5.2 YOLOv5 Student: Teacher Labels Only

Unfortunately, since this model was only trained on the teacher's pseudo-labels with no modifications aside from the backfill (which was required in order to compensate for the classes that were missing on the pseudo-labels), it performed slightly worse than the baseline model. The model still overpredicts background for most GT boxes, dragging recall and the mAP metrics down with it. At the 25th epoch, the student model reached a pitiful 0.0927 recall score: down from 0.113 at the same epoch for the baseline model. The mAP@0.5 score for the student model at that same epoch was 0.085, also down from the baseline model's 0.0973 for the same metric. Therefore, across all metrics, the student model performed worse than the baseline at the same epoch. You may observe Figure 7 for a visualization of the results for this category.

FIGURE 7: YOLOv5 STUDENT RESULTS (TEACHER'S LABELS ONLY)



5.3 YOLOv5 Student: OWLv2 Refinements

It is clear from many examples that the Open Vocabulary Detection model OWLv2 by Google DeepMind is satisfactory in its performance with refining bounding boxes drawn by the YOLOv5 baseline model. Almost always, OWLv2 will draw a more exact box over the object. The point of this step was to potentially increase mAP@0.5 and possibly also mAP@0.5:0.95, seeing as YOLOv5 is good at drawing the boxes over the objects but oftentimes are not exact.

Figure 8 shows us the teacher's bounding boxes over the objects (in red), and the bounding boxes that were exported (in green). If a bounding box is exported, it has either been kept or refined by OWLv2. As is visible from the image, OWLv2 will export bounding boxes that fit more snugly around the object, as opposed to the teacher's that may cover a more significant amount of space.

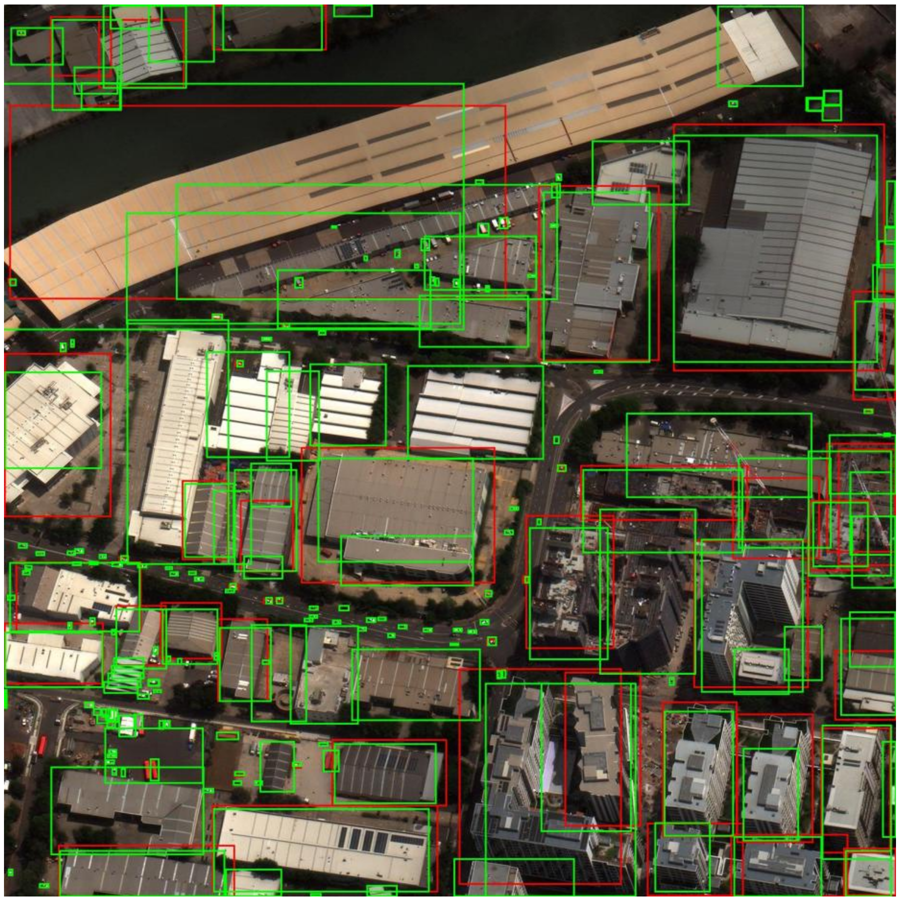
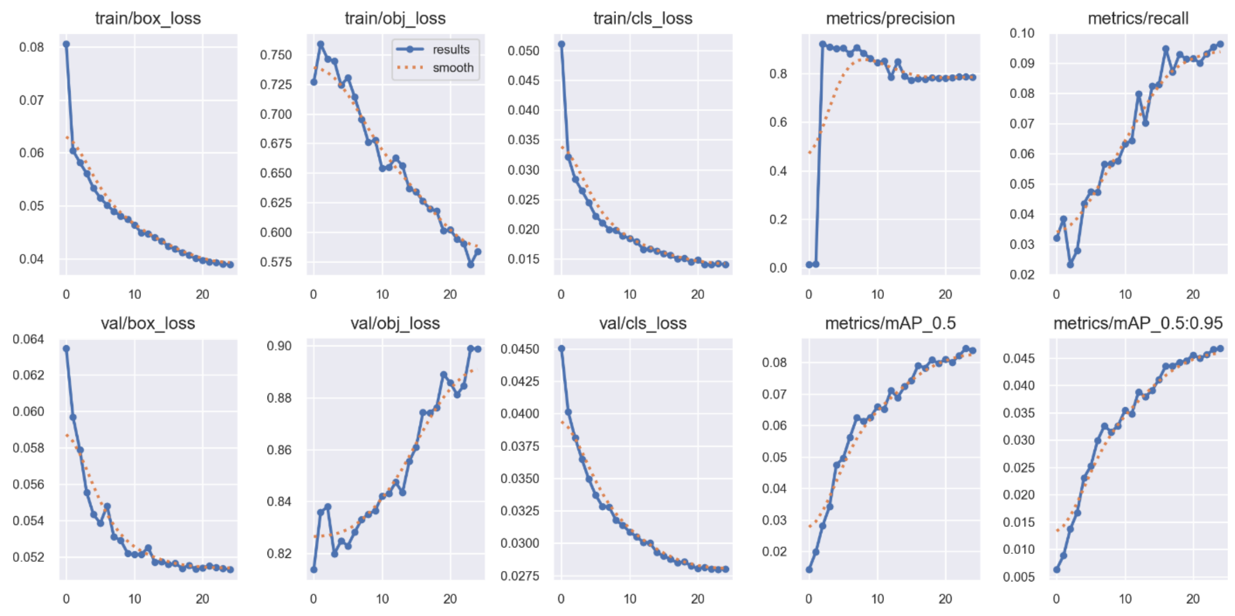


FIGURE 8: REFINEMENTS BY OWLv2

*Teacher's boxes in Red, Owlv2's exported boxes in green (whether resized or just confirmed.

However, what we know here to be rather satisfactory refinements by the OWLv2 model, do not seem to have much of an effect on the performance of the student model that was trained on its respective OWLv2-curated dataset. The only metric that is known to have improved through OWLv2 refinement is recall, and the improvement was only marginal. Moreover, the metric that we were trying to improve, mAP@0.5 and mAP@0.5:0.95, actually showed a worse performance than the baseline model. This may be attributed to our choice of parametrization, which forced the OWLv2 model to be more conservative and careful about the boxes it refines. Figure 9 shows more of the same that we've seen from the other student model and the baseline model.

FIGURE 9: OWLv2 REFINEMENT-ONLY RESULTS



Another possible improvement in this area would have been to allow OWLv2 to add novel objects and possibly filter out boxes. This option was discarded previously due to the intuition that OWLv2 is not properly tuned to handle objects as seen from overhead, but this could be an opportunity for further study.

FIGURE 10

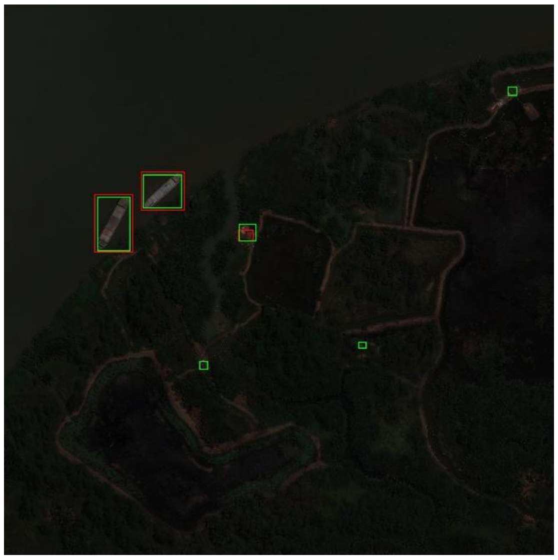


FIGURE 11

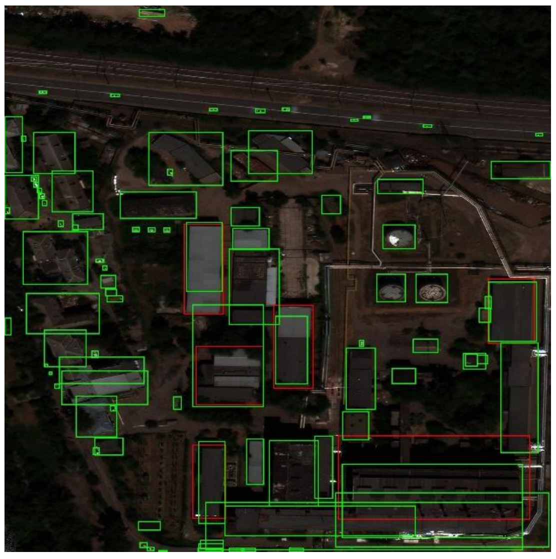


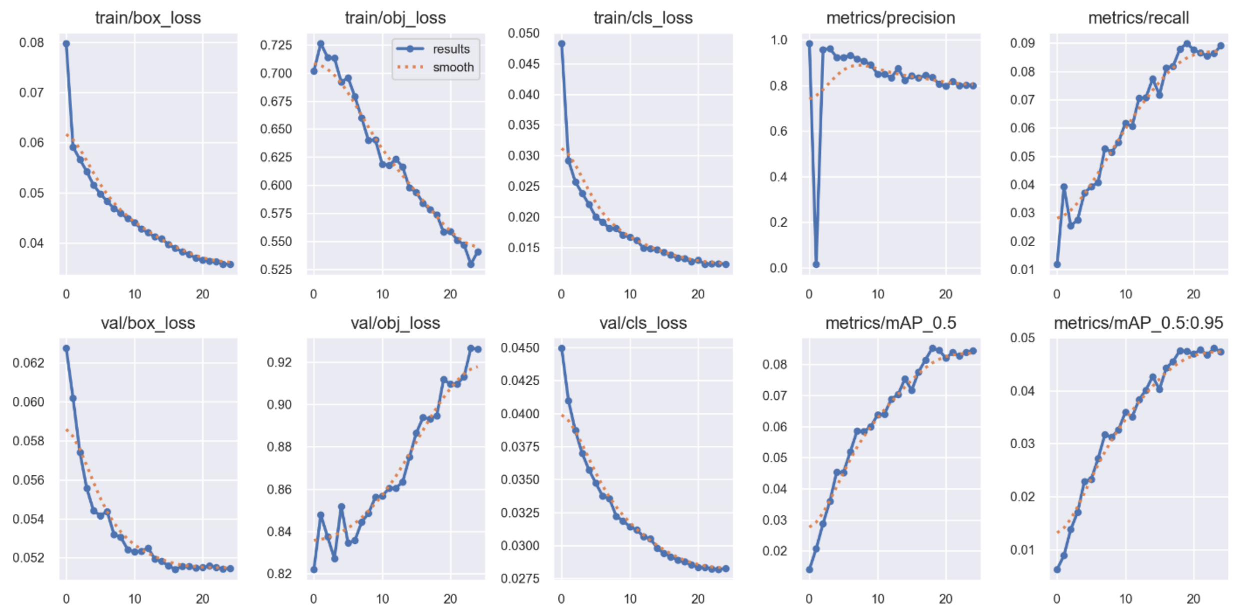
Figure 11 demonstrates that while OWLv2 is generally able to draw bounding boxes that are superior to those drawn by YOLOv5, it can be faulty (refer to the object at the lower left of center) and it is possible that in a significant portion of drawn bounding boxes that the exported bounding boxes (the ones in green) are actually worse. If this is the case, which we know for a fact does happen in both Figure 10 and Figure 11, whatever improvement there may have been in mAP metrics by the superior boxes could be canceled out by a significant number of inferior ones. It is hypothesized that these are important reasons as to why the performance of the OWLv2 refined student model on the training set is similar to that of the student model trained only on the teacher's pseudo-labels.

5.4 RS-CLIP False Positive Filtering

This filtering step by CLIP was intended to have clip drop a good number of pseudo-labels based on semantic similarity. The intuition for including CLIP into this pipeline was to trim the number of FPs the model may have made by dropping a good portion of bounding boxes the YOLOv5 baseline model could have left behind, thereby making the dataset slightly less noisy for the student model. The issue with this is that throughout our experimentation, even when CLIP is nudged toward viewing objects from overhead, it still underperforms in terms of being able to know whether that specific object matches its class name.

Many of the other VLMs we tested for this task, including ones like BLIP-2 and Qwen, underperform in their ability to describe very small objects, given that most of them are from an overview perspective and are only shown in just a few pixels. It is no surprise that these objects are difficult in general to recognize, even to the human eye at times. The purpose was to eliminate some bounding boxes, but not too many; in the hope that most of them are actually FPs and that we are increasing generalization and performance by eliminating them. Figure 12 demonstrates that this step did not meaningfully help model performance and may even have worsened it compared to the OWLv2 refinement step.

FIGURE 12: RESULTS FOR CLIP FILTERING OF BASELINE PSEUDO-LABELS ONLY



It is clear from Figure 12 that after the 20th epoch, recall plateaued and even dropped below the 0.09 mark we had achieved for the previous steps. As for the rest of the metrics, the trend we see consistently where precision drops as recall increases is present, as is the low performance in general.

5.5 RS-CLIP and OWLv2 processed pseudo-labels

This step is intended to merge the refinement capabilities of OWLv2 and the filtering capabilities of OpenCLIP. Intuitively, it is recognized that if we refine YOLOv5 bounding boxes with better ones, namely ones drawn more accurately over an object, especially after cutting FPs like background tiles with no objects in them sounds plausible, defensible, and implementable. We have seen that OWLv2 is capable of refining inaccurate boxes around objects well enough, and it speaks to its capacity to recognize that there is in fact an object inside whatever bounding box it is given. The same cannot be said of CLIP, where we rely on it to recognize whether it can properly associate and visually ground the detected object to the class names, especially with the help of the prompts.

CLIP results in the training phase show that this may not be the case, and it is entirely possible that CLIP may not be dropping boxes because the class names are not semantically compatible with the object, but more so because there is no object to ground in the semantic information of a class name.

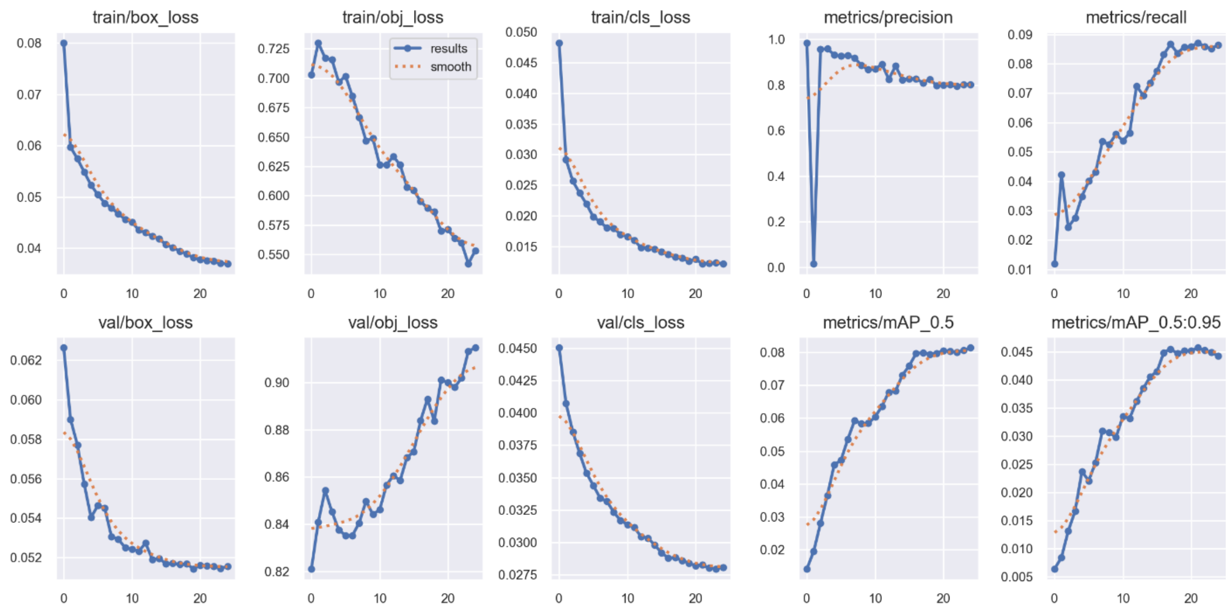


FIGURE 13: RESULTS AFTER OWLv2 REFINEMENT AND CLIP FILTERING

During the training phase, what intuitively could be the best performing setup for the post-processing of the pseudo-labels, the framework performed slightly worse than the CLIP filtering step, which means that it makes this setup the worst performing in our experimentation. Plateauing and decrease in performance metrics such as recall and the mAP@0.5 metrics is clear, and we can even notice that precision stops decreasing, and plateaus just as recall also plateaus and slightly decreases.

6 – Test Results, Error Analysis and Evaluation

6.1 Overview and comparison of test results

All in all, the YOLOv5 model trained on the raw GT labels (as in, the original, tiled dataset) performed the best out of all the student models; no student model was able to surpass the performance of its teacher model overall. Observe Figure 14, which represents the overall scores in all metrics for every model after running them on the xView test set; from the baseline all the way down to the last student model.

*Note: The baseline model was trained for 50 epochs, while the students for 25 epochs each.

	Precision	***Recall***	***mAP@0.5***	***mAP@0.5:0.95***
YOLOv5 Baseline	0.578	0.196	0.157	0.083
YOLOv5 Student	0.771	0.122	0.103	0.0524
YOLOv5 OWLv2-only	0.759	**0.126**	**0.106**	**0.0528**
YOLOv5 CLIP-only	0.788	0.101	0.102	0.0515
YOLOv5 hybrid	0.784	0.105	0.103	0.0514

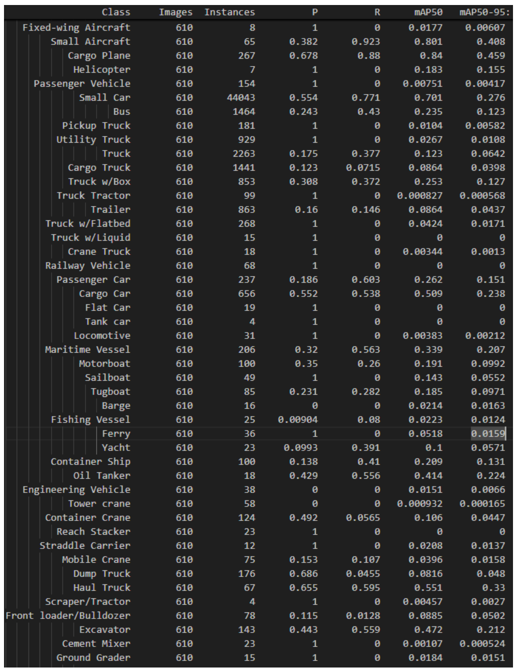
FIGURE 14: YOLOv5 RESULTS WHEN RUN ON XVIEW TEST SET

It is imperative to remember that the student models having higher precision does not signify that they perform better than the baseline, rather that they may be too conservative and over-regularized, or too constricted by their parameters. It also means that they are not making many predictions, but most of the predictions they are making are accurate. From the outset, the most obvious difference in the performance of the different student models, the OWLv2-only model boasts the highest recall, mAP@0.5, and mAP0.5:0.95. This is likely due to the fact that the OWLv2 model improved box localization, which allowed the student to better understand, even if only slightly, more discrete characteristics. By also allowing OWLv2 to just verify the bounding boxes and refine them if needed, we avoided losing valuable material that would have otherwise been filtered away. We hypothesize that this is what caused the CLIP model to underperform, the elimination of not only many false positives, but also true negatives that it disagreed with.

Finally, when observing the performance scores for the hybrid OWLv2-CLIP-trained student model, it is clear that the scores for each even out; CLIP causing a decrease while OWLv2 an increase. This speaks to the usefulness of using OVODs in zero-shot contexts such as this one. Even if the differences in performance were not necessarily too different, we can still extrapolate which step is best when working on an object detection pipeline that employs the help of VLMs.

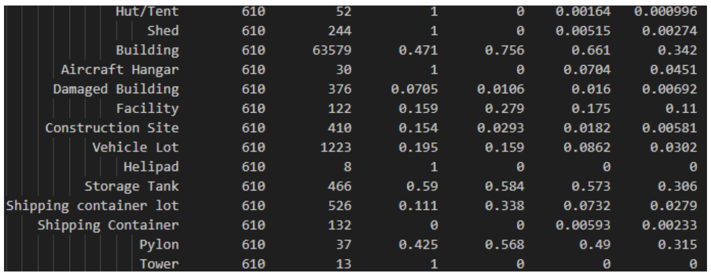
6.2 A closer look at test results: per-class scores

FIGURE 15



Class	Images	Instances	P	R	mAP50	mAP50-95:
Fixed-wing Aircraft	610	8	1	0	0.0177	0.00607
Small Aircraft	610	65	0.382	0.923	0.801	0.408
Cargo Plane	610	267	0.678	0.88	0.84	0.459
Helicopter	610	7	1	0	0.183	0.155
Passenger Vehicle	610	154	1	0	0.00751	0.00417
Small Car	610	44043	0.554	0.771	0.701	0.276
Bus	610	1464	0.243	0.43	0.235	0.123
Pickup Truck	610	181	1	0	0.0104	0.00582
Utility Truck	610	929	1	0	0.0267	0.0108
Truck	610	2263	0.175	0.377	0.123	0.0642
Cargo Truck	610	1441	0.123	0.0715	0.0864	0.0398
Truck w/Box	610	853	0.308	0.372	0.253	0.127
Truck Tractor	610	99	1	0	0.000827	0.000568
Trailer	610	863	0.16	0.146	0.0864	0.0437
Truck w/Flatbed	610	268	1	0	0.0424	0.0171
Truck w/Liquid	610	15	1	0	0	0
Crane Truck	610	18	1	0	0.00344	0.0013
Railway Vehicle	610	68	1	0	0	0
Passenger Car	610	237	0.186	0.603	0.262	0.151
Cargo Car	610	656	0.552	0.538	0.509	0.238
Flat Car	610	19	1	0	0	0
Tank car	610	4	1	0	0	0
Locomotive	610	31	1	0	0.00383	0.00212
Maritime Vessel	610	206	0.32	0.563	0.339	0.207
Motorboat	610	100	0.35	0.26	0.191	0.0992
Sailboat	610	49	1	0	0.143	0.0552
Tugboat	610	85	0.231	0.282	0.185	0.0971
Barge	610	16	0	0	0.0214	0.0163
Fishing Vessel	610	25	0.00904	0.08	0.0223	0.0124
Ferry	610	36	1	0	0.0518	0.0159
Yacht	610	23	0.0993	0.391	0.1	0.0571
Container Ship	610	100	0.138	0.41	0.209	0.131
Oil Tanker	610	18	0.429	0.556	0.414	0.224
Engineering Vehicle	610	38	0	0	0.0151	0.0066
Tower crane	610	58	0	0	0.000932	0.000165
Container Crane	610	124	0.492	0.0565	0.106	0.0447
Reach Stacker	610	23	1	0	0	0
Straddle Carrier	610	12	1	0	0.0208	0.0137
Mobile Crane	610	75	0.153	0.107	0.0396	0.0158
Dump Truck	610	176	0.686	0.0455	0.0816	0.048
Haul Truck	610	67	0.655	0.595	0.551	0.33
Scraper/Tractor	610	4	1	0	0.00457	0.0027
Front loader/Bulldozer	610	78	0.115	0.0128	0.0885	0.0502
Excavator	610	143	0.443	0.559	0.472	0.212
Cement Mixer	610	23	1	0	0.00107	0.000524
Ground Grader	610	15	1	0	0.0184	0.0151

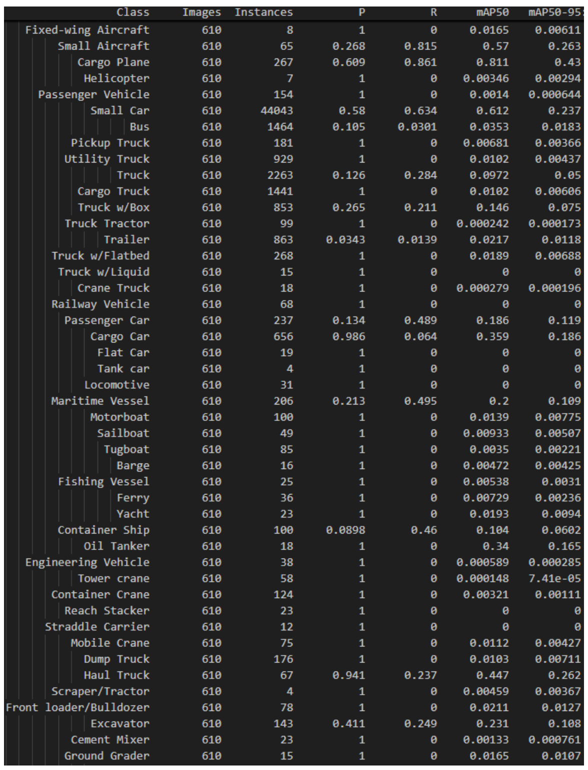
FIGURE 16



Hut/Tent	610	52	1	0	0.00164	0.000996
Shed	610	244	1	0	0.00515	0.00274
Building	610	63579	0.471	0.756	0.661	0.342
Aircraft Hangar	610	30	1	0	0.0704	0.0451
Damaged Building	610	376	0.0705	0.0106	0.016	0.00692
Facility	610	122	0.159	0.279	0.175	0.11
Construction Site	610	410	0.154	0.0293	0.0182	0.00581
Vehicle Lot	610	1223	0.195	0.159	0.0862	0.0302
Helipad	610	8	1	0	0	0
Storage Tank	610	466	0.59	0.584	0.573	0.306
Shipping container lot	610	526	0.111	0.338	0.0732	0.0279
Shipping Container	610	132	0	0	0.00593	0.00233
Pylon	610	37	0.425	0.568	0.49	0.315
Tower	610	13	1	0	0	0

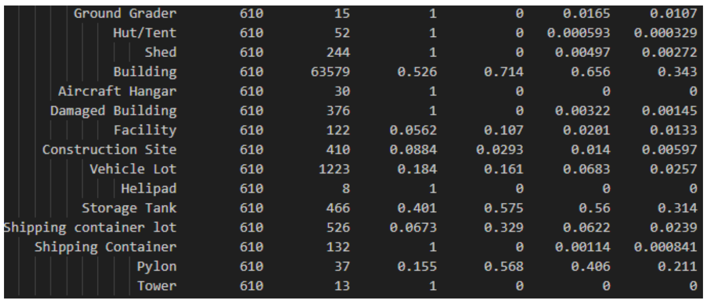
Figures 15 and 16 show us every single one of the metrics per class for the YOLOv5 baseline model. The baseline model gave us a recall score of 0 for ~29 classes. This directly explains the pitiful overall recall score, and mAP@0.5 by extension. This essentially means that all of the detections made for that class were false positives. It is also no doubt that this is due to the vanishingly small number of instances of those classes, especially when there are some classes that have thousands if not tens of thousands of instances (see Tank Car, Scraper/Tractor, Locomotive, etc.). It is also clear that the model performs best when there are at least >100 instances for the model to learn from (see Building, Small Car, Excavator, Dump Truck, etc.).

FIGURE 17



```
                 Class     Images  Instances          P          R      mAP50   mAP50-95
   Fixed-wing Aircraft        610          8          1          0     0.0165    0.00611
        Small Aircraft        610         65      0.268      0.815       0.57      0.263
           Cargo Plane        610        267      0.609      0.861      0.811       0.43
            Helicopter        610          7          1          0    0.00346    0.00294
     Passenger Vehicle        610        154          1          0     0.0014   0.000644
             Small Car        610      44043       0.58      0.634      0.612      0.237
                   Bus        610       1464      0.105     0.0301     0.0353     0.0183
          Pickup Truck        610        181          1          0    0.00681    0.00366
         Utility Truck        610        929          1          0     0.0102    0.00437
                 Truck        610       2263      0.126      0.284     0.0972       0.05
           Cargo Truck        610       1441          1          0     0.0102    0.00606
           Truck w/Box        610        853      0.265      0.211      0.146      0.075
         Truck Tractor        610         99          1          0   0.000242   0.000173
               Trailer        610        863     0.0343     0.0139     0.0217     0.0118
       Truck w/Flatbed        610        268          1          0     0.0189    0.00688
        Truck w/Liquid        610         15          1          0          0          0
           Crane Truck        610         18          1          0   0.000279   0.000196
       Railway Vehicle        610         68          1          0          0          0
         Passenger Car        610        237      0.134      0.489      0.186      0.119
             Cargo Car        610        656      0.986      0.064      0.359      0.186
              Flat Car        610         19          1          0          0          0
              Tank car        610          4          1          0          0          0
            Locomotive        610         31          1          0          0          0
       Maritime Vessel        610        206      0.213      0.495        0.2      0.109
             Motorboat        610        100          1          0     0.0139    0.00775
              Sailboat        610         49          1          0    0.00933    0.00507
               Tugboat        610         85          1          0     0.0035    0.00221
                 Barge        610         16          1          0    0.00472    0.00425
        Fishing Vessel        610         25          1          0    0.00538     0.0031
                 Ferry        610         36          1          0    0.00729    0.00236
                 Yacht        610         23          1          0     0.0193     0.0094
        Container Ship        610        100     0.0898       0.46      0.104     0.0602
            Oil Tanker        610         18          1          0       0.34      0.165
   Engineering Vehicle        610         38          1          0   0.000589   0.000285
           Tower crane        610         58          1          0   0.000148   7.41e-05
       Container Crane        610        124          1          0    0.00321    0.00111
         Reach Stacker        610         23          1          0          0          0
      Straddle Carrier        610         12          1          0          0          0
          Mobile Crane        610         75          1          0     0.0112    0.00427
            Dump Truck        610        176          1          0     0.0103    0.00711
            Haul Truck        610         67      0.941      0.237      0.447      0.262
       Scraper/Tractor        610          4          1          0    0.00459    0.00367
Front loader/Bulldozer        610         78          1          0     0.0211     0.0127
             Excavator        610        143      0.411      0.249      0.231      0.108
          Cement Mixer        610         23          1          0    0.00133   0.000761
         Ground Grader        610         15          1          0     0.0165     0.0107
```

FIGURE 18



```
         Ground Grader        610         15          1          0     0.0165     0.0107
              Hut/Tent        610         52          1          0   0.000593   0.000329
                  Shed        610        244          1          0    0.00497    0.00272
              Building        610      63579      0.526      0.714      0.656      0.343
       Aircraft Hangar        610         30          1          0          0          0
      Damaged Building        610        376          1          0    0.00322    0.00145
              Facility        610        122     0.0562      0.107     0.0201     0.0133
     Construction Site        610        410     0.0884     0.0293      0.014    0.00597
           Vehicle Lot        610       1223      0.184      0.161     0.0683     0.0257
               Helipad        610          8          1          0          0          0
          Storage Tank        610        466      0.401      0.575       0.56      0.314
Shipping container lot        610        526     0.0673      0.329     0.0622     0.0239
    Shipping Container        610        132          1          0    0.00114   0.000841
                 Pylon        610         37      0.155      0.568      0.406      0.211
                 Tower        610         13          1          0          0          0
```

Immediately the first obvious thing is the number of whole classes of objects that were completely mislabeled by the YOLOv5 Teacher-only student model. ~40 classes have a 0 for a recall value. This unfortunately demonstrates that the student models are even worse than the baseline when it comes to detecting objects for which there are a small number of instances. Since mAP@0.5 also accounts for recall values, its value as a metric is relatively tied to it and is therefore also dragged down by the sheer number of false positives. The OWLv2-only model shares the same number of classes in which recall is given a value of 0, ~40.

Is it also important to note that for most of these classes for which a recall value of 0 was given, that the majority of them did have a mAP@0.5 and mAP@0.95 score. This means that the objects were indeed detected, but they were misclassified. For instance, for the OWLv2-only test results, around 40 classes were indeed completely misclassified, but only ~8 of them have no mAP value whatsoever. Without fail, all of the classes of objects that did not have a value for recall or mAP had, at most, 68 instances (for the Railway Vehicle class: other classes like Reach Stacker and Aircraft Hanger had 23 and 30 respectively; others, even less).

When we look at the values for CLIP-only and the Hybrid approach, the number of classes without a recall value grows to ~42, indicating that some classes were indeed filtered out when CLIP decided they were false positives. This indicates that we need a more robust approach when using models like CLIP that use semantic knowledge in order to more accurately filter out false positives.

6.3 Visual examples & Error Analysis

The YOLOv5 models, across all categories, exported quite a significant number of false positives themselves. Often times not making any sense at all. Figure 19 demonstrates a very noisy and overwhelming image that shows perfectly the sheer amount of noise that the YOLOv5 Teacher-only student model is struggling with. Figure 20 represents the baseline model's detections and is put beside its direct student model to properly appreciate how poorly the student deals with the extra noise and prevents it from learning the characteristics of different objects more accurately. We can see that anywhere from random patches of land being labeled as construction sites and aircraft hangars being detected as many separate buildings instead, the YOLOv5 student is detecting things even when there is absolutely nothing to be found. While the pictures are not the same, they contain a similar enough scenario that they can be faithfully compared.

FIGURE 19

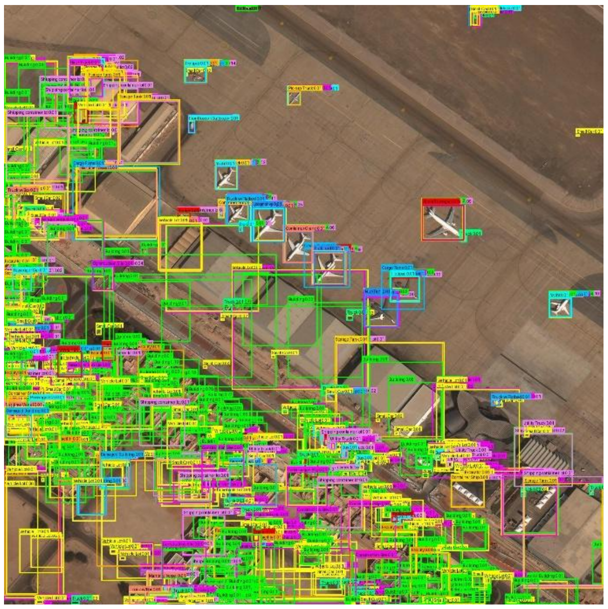
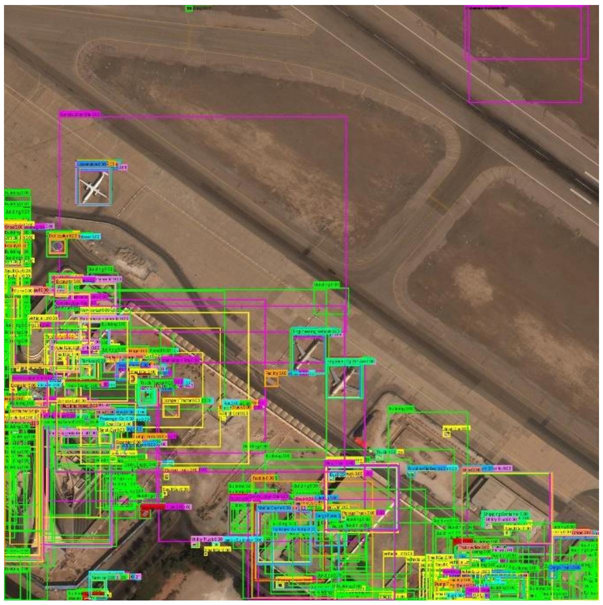


FIGURE 20



From this particular qualitative, visual comparison, we can extrapolate that the extra noise introduced by the baseline model into its pseudo-labels not only did not help the YOLOv5 student models to better generalize, the students learned and practiced the teacher model's mistakes. In order words, the extra noise integrated by the baseline model misled all of the students to the point that very little could be done in terms of semantic information to improve the situation.

7 – **Reflection on the project and Conclusion**

Even if the framework that was used to test whether VLMs or semantic information could be used to ameliorate model performance demonstrated poor performance, we believe that this did not eliminate the merit and outcome of the experiments and actually gave us some important information. For instance, we know for a fact that using OVODs like OWLv2, even if only conservatively, and for auditing tasks and fixing bounding boxes to more accurate ones by using class-aware prompts to allow the model to make these decisions, is indeed useful for the purpose of improving the outcome and quality of pseudo-labels that may be generated by a baseline model. Furthermore, the literature review section presents a compelling argument for the use of OVODs to not only audit but also contribute to the quality of the pseudo-labels by introducing its own detections through zero-shot learning; this could very well have helped increase the number of TNs at the very least. In terms of the approach to filtering FPs using VLMs such as CLIP, the implementation of more robust and complex systems is needed; beyond simple class-aware prompting to nudge the model towards remote sensing object classification. Using VLMs for FP filtering in image classification is difficult, especially when relating to remote sensing tasks. We carried out some experimentation while in the process of building this pipeline that proved to us that VLM captions are extremely unreliable when asking even massive and robust VLMs to produce captions for objects that are especially small and only a few pixels in size, not to mention that this is also within the remote sensing task, which makes it all the more difficult and unreliable.

REFERENCES

Akhyar, A. et al. 2024. Deep artificial intelligence applications for natural disaster management systems: A methodological review. *Ecological Indicators* 163, 112067. doi: 10.1016/j.ecolind.2024.112067

Caron, M., Fathi, A., Schmid, C. and Iscen, A. 2024. Web-scale visual entity recognition: An LLM-driven data approach. *arXiv preprint* arXiv:2410.23676. Available at: https://arxiv.org/abs/2410.23676 [Accessed: 8 September 2025]

Gao, M., Xing, C., Niebles, J.C., Li, J., Xu, R., Liu, W. and Xiong, C. 2022. Open vocabulary object detection with pseudo bounding-box labels. *arXiv preprint* arXiv:2111.09452. Available at: https://arxiv.org/abs/2111.09452 [Accessed: 8 September 2025]

Li, L.H., Yatskar, M., Yin, D., Hsieh, C.-J. and Chang, K.-W. 2019. VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint* arXiv:1908.03557. Available at: https://arxiv.org/abs/1908.03557 [Accessed: 8 September 2025]

Liu, S. et al. 2024. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv preprint* arXiv:2303.05499. Available at: https://arxiv.org/abs/2303.05499 [Accessed: 8 September 2025]

Minderer, M., Gritsenko, A. and Houlsby, N. 2024. Scaling open-vocabulary object detection. *arXiv preprint* arXiv:2306.09683. Available at: https://arxiv.org/abs/2306.09683

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. 2016. You only look once: Unified, real-time object detection. *arXiv preprint* arXiv:1506.02640. Available at: https://arxiv.org/abs/1506.02640 [Accessed: 8 September 2025]

Radford, A. et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint* arXiv:2103.00020. Available at: https://arxiv.org/abs/2103.00020 [Accessed: 8 September 2025]

Xie, Q., Luong, M.-T., Hovy, E. and Le, Q.V. 2020. Self-training with Noisy Student improves ImageNet classification. *arXiv preprint* arXiv:1911.04252. Available at: https://arxiv.org/abs/1911.04252 [Accessed: 8 September 2025]

Zhu, C. and Chen, L. 2024. A survey on open-vocabulary detection and segmentation: Past, present, and future. *arXiv preprint* arXiv:2307.09220. Available at: https://arxiv.org/abs/2307.09220 [Accessed: 8 September 2025]

Chen, Y., Wang, Q. and Yan, S. 2025. A survey on vision-language models and object detection. *International Journal of Modern Research in Engineering and Technology* 10(3). Available at: https://www.ijmret.org [Accessed: 8 September 2025]

Jegham, N., Koh, C.Y., Abdelatti, M. and Hendawi, A. 2025. YOLO evolution: A comprehensive benchmark and architectural review of YOLOv12, YOLO11, and their previous versions. *arXiv preprint* arXiv:2411.00201. Available at: https://arxiv.org/abs/2411.00201 [Accessed: 8 September 2025]

Lee, D.-H. 2013. Pseudo-Label: The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop: Challenges in Representation Learning (WREPL)*. Atlanta, GA, USA. Available at: https://www.researchgate.net/publication/280581078_Pseudo-Label_The_Simple_and_Efficient_Semi-Supervised_Learning_Method_for_Deep_Neural_Networks [Accessed: 8 September 2025]

Li, X., Wen, C., Hu, Y. and Zhou, N. 2023. RS-CLIP: Zero-shot remote sensing scene classification via contrastive vision-language supervision. *International Journal of Applied Earth Observation and Geoinformation* 124, 103497. doi: 10.1016/j.jag.2023.103497.

Lu, H., Bian, Y. and Shah, R.C. 2025. CLIPGRADER: Leveraging vision-language models for robust label quality assessment in object detection. *arXiv preprint* arXiv:2503.02897. Available at: https://arxiv.org/abs/2503.02897 [Accessed: 8 September 2025]

Sapkota, R. and Karkee, M. 2025. Object detection with multimodal large vision-language models: An in-depth review. *arXiv preprint* arXiv:2508.19294. Available at: https://arxiv.org/abs/2508.19294 [Accessed: 8 September 2025]