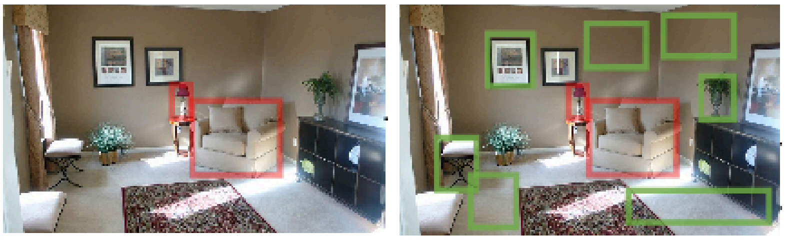
# Computer Vision - Term Project Report - Group 30

Exploiting Vision-Language Models for OWOD

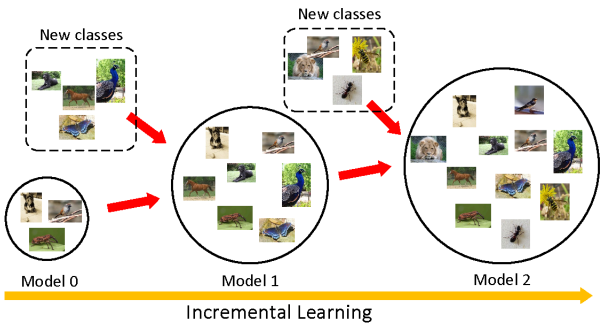
312511814 Tom Broudin, 312553002 蘇煒閎,
312605013 張家倫, 312605024 戴麒恩

## 1. Open World Object Detection

- Our project will tackle the challenge of **Open World Object Detection**, which is to **detect unknown objects** that are undefined in the training dataset. Once the computer detects what seem to be objects within a given image, the model is expected to **locate and highlight** the objects that do not belong to the defined class subset of the training dataset.



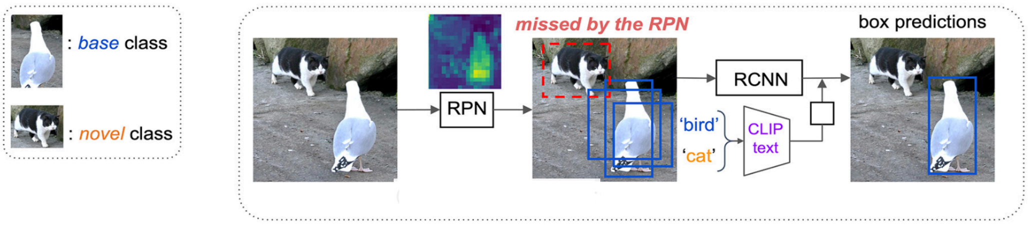
- **Incremental learning** is another aspect of OWOD. Once the model has been trained on a particular dataset and that unknown objects have been detected, we can further add those unknown objects to our current training subset to increase the detection model's knowledge.



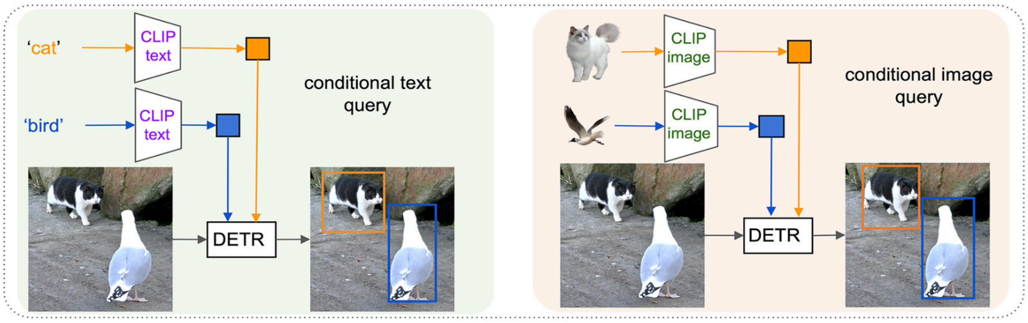

## 2. Open Vocabulary Detection (OVD)

- **Open Vocabulary Detection** aims to enable object detection models to identify and localize objects that were not seen during the training phase with the integration of **language knowledge** expanding the capability of traditional object detection systems, which are limited to detecting a fixed set of predefined classes.
- **OVD's key features:**
  **1.Generalization**: Ability to detect novel object categories beyond those seen during training.
  **2.Scalability**: Ability to handle a large and dynamically changing set of object categories.
  **3.Flexibility**: Ability to adapt to new objects and categories without needing extensive retraining.

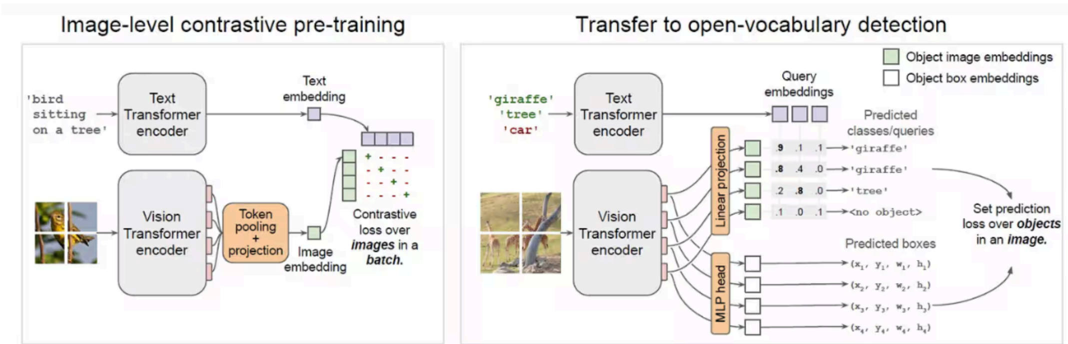
- **Comparison between a RPN-based detector and OVD**
  The RPN trained on **closed-set** object classes tends to ignore novel classes (e.g., the `cat' region receives little response). Hence the cats in this example are largely missed with few to no proposals.




  By contrast, **OVD** is trained to perform matching between a conditional query and its corresponding box, which helps the learning of the **text-image correspondence** that can generalize queries to unseen classes.
  It can take input queries in the form of either text (class name like cat or bird) or images.

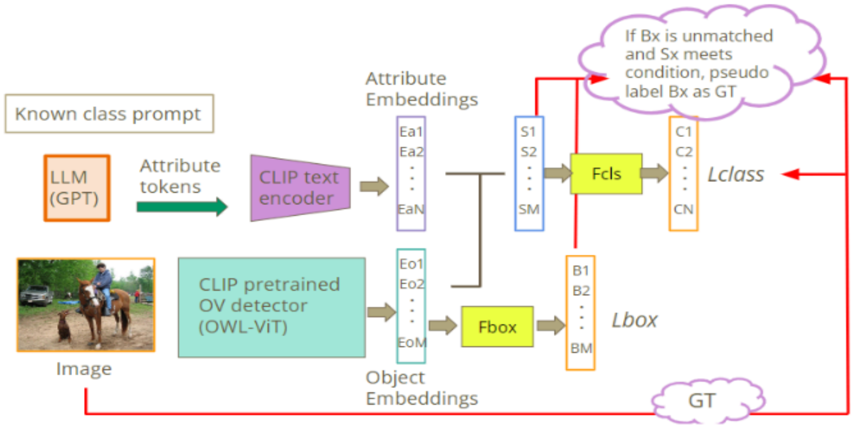



- An OVD method: **OWL-ViT(Open-World Localization with Vision Transformers)**
  OWL-ViT includes **two steps** in its training strategy:
  1. Pre-train an **image** and **text encoder** contrastively using image-text pairs, similar to CLIP.
  2. To achieve OVD, query strings are embedded with the text encoder and used for classification. We can classify objects with the scores. Compared to the first part, the contrast only indicates positive & negative. The model is **fine-tuned on standard detection datasets**. We can use text-derived embeddings for OVD, or image-derived embeddings as queries.

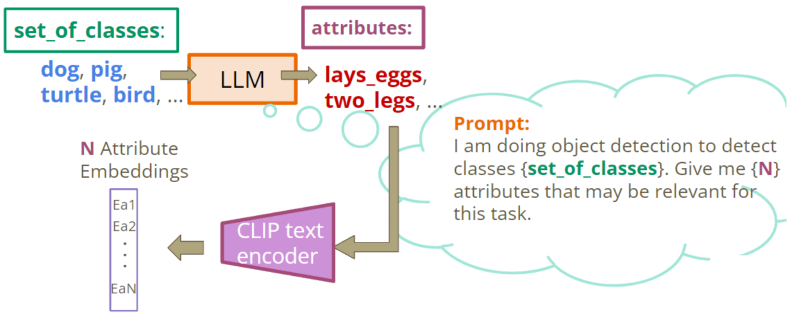


## 3. Method

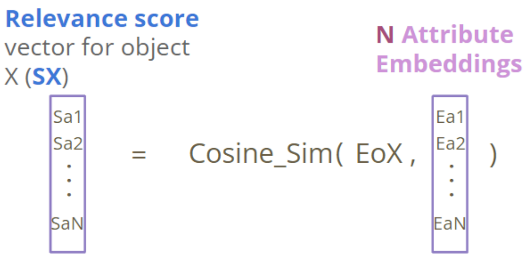
The figure summarizes our method, which aims to **incorporate vision-language knowledge into the scenario of open world object detection**. The overall framework involves **Large Language Models**, **CLIP**, and a CLIP pretrained open vocabulary object detector (In this case, **OWL-ViT**). The main intuition behind this is to exploit the correlation between vision and language embeddings to find potential **objects of interest** given the currently known set of classes and their mutual or distinctive **attributes**, and to give the detection model the ability to predict objects out of the known set, by labeling these discovered potential objects as **pseudo ground truth labels** during training.



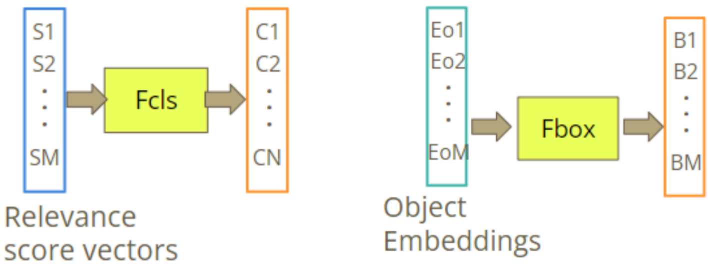

- In an open world object detection task, we are given a set of classes that are present as ground truth labels in the training data. We feed this set of classes to a large language model with a prompt, which **asks the language model what attributes we would be interested in** if we were to train an object detection model on these classes. When we have these attributes from the LLM, we can further feed them into a **CLIP text encoder** to obtain the **embeddings** of these attributes.




- For the object detector we intentionally choose one that is **pretrained with language knowledge**, so that the embeddings it generates can be conveniently linked to the text embeddings of the attributes. Each embedding produced by the detector (EoX) can be seen as an **object proposal**, we compare the **similarity of each of these embeddings with the attribute embeddings** from before to see how well it fits the attributes. In practice, for every object, we calculate the cosine similarity between the object embedding and every attribute embedding to obtain a **relevance score vector** for that object proposal. Each entry in this relevance score vector represents the relevance of the object and a particular attribute. For example, Sa1in SX is the relevance of EoX and Ea1, SaN in SX is the relevance of EoX and EaN.




- To convert the vision embeddings from the detection model to actual objects, we have to design a **classification head** and a **box localization head**. These heads are feed forward networks that convert the inputs to a desired format, such as **class probability vectors for the classification head** and a 4 element vector that specifies the coordinates of the **predicted boxes for the box localization head**. One difference here in our design from other object detectors is that the **input of the classification head is changed to the relevance score vectors of the objects**, since we reckon that the object's relations to the attributes may be a more straightforward indicator to the distinctiveness between classes. We also use a **weight matrix in replacement of the neural network** in order to track the weights for each attribute more straightforwardly.




- After the object embeddings are passed through the box localization head, we can **match their box locations with those in the training data** to see which of them correspond to the ground truth labels. For each **unmatched box**, we check whether its **relevance score passes the criteria in the purple box**, which indicates whether it is strongly related to a defined attribute.
An object **passing the test** means that it indicates some strong similarity to the attributes shared in the known class set but is not one of the the defined classes.These objects are considered as **likely unknown objects** and can thus be further treated as **pseudo labels** of the "unknown class" in the loss calculation stage of each iteration, encouraging the model to learn to detect objects out of the known set.

If
***Max*** ( ***Sigmoid***( SX ) ) > Threshold

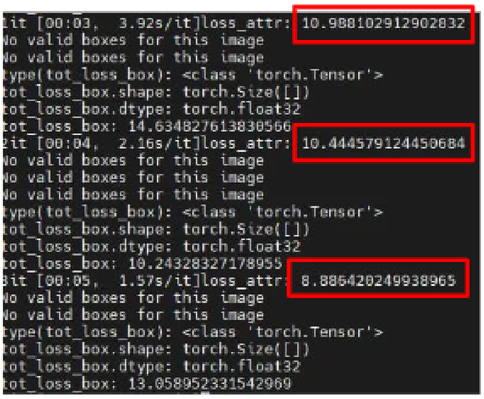
Treat X as a GT object of class
"unknown" in loss calculation

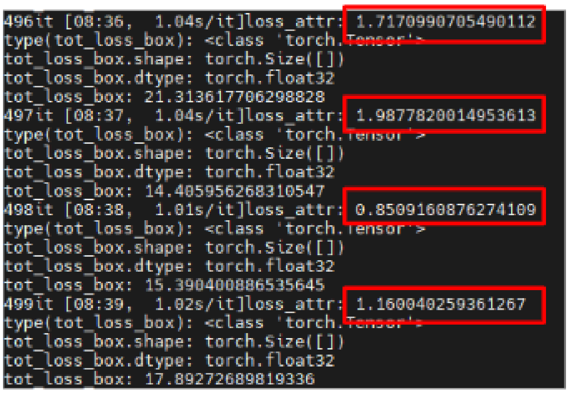
During **inference time**, we examine the **output logits** of each embedding. If the **class logits** of an embedding does not show signs of strong confidence but the **attribute relevance vector** shows that the object has high values with some of the defined attributes, we **postprocess that prediction as an**

**unknown prediction**.

## 4. Results

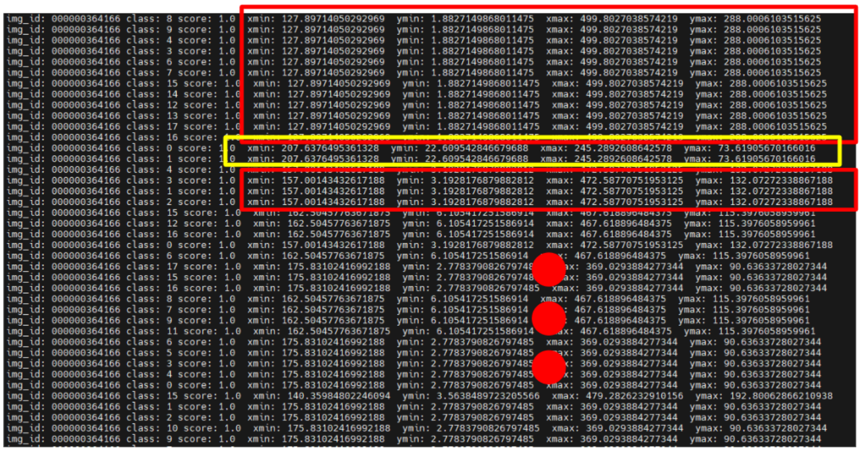
### Validness of utilizing text-vision correspondence for classification:




The two figures above are respectively the training output logs of **early stages** in an epoch and **later stages** in an epoch. As one can observe, the **loss_attr output**, which is calculated by **Cross Entropy loss** between GT labels and matched predictions, lowers significantly throughout training. This indicates that **using the relevance vector of a vision embedding and attribute embeddings as class head input is a valid method for classification**.

### Inference Predictions:



We design the model to output its **top 100 most confident predictions for each image** during inference. However, printing out the predictions shows that multiple **predictions have the exact same bounding boxes**, which severely diminishes the performance of the model. **We were unfortunately unable to identify the cause of this due to the limitation of time**.

```
Test: Total time: 0:06:57 (0.8420 s / it)
aeroplane has 32249 predictions.
0
bicycle has 19169 predictions.
316
bird has 36420 predictions.
440
boat has 45609 predictions.
430
bus has 16497 predictions.
285
car has 26307 predictions.
1932
cat has 18669 predictions.
202
cow has 28603 predictions.
380
dog has 23309 predictions.
218
horse has 33923 predictions.
273
motorbike has 18374 predictions.
0
sheep has 36448 predictions.
361
train has 22778 predictions.
190
elephant has 24110 predictions.
255
bear has 50054 predictions.
71
zebra has 34080 predictions.
268
giraffe has 18702 predictions.
232
truck has 9899 predictions.
415
person has 0 predictions.
11004
```

```
Current class AP50: tensor(0.0336)
Current class Precisions50: 0.16194336076233884
Current class Recall50: nan
Known AP50: tensor(0.0336)
Known Precisions50: 0.16194336076233884
Known Recall50: nan
Unknown AP50: tensor(0.)
Unknown Precisions50: 0.0
Unknown Recall50: 0.0
aeroplane 0.000000
bicycle 0.008494
bird 0.003798
boat 0.006853
bus 0.061671
car 0.200777
cat 0.042853
cow 0.007517
dog 0.029043
horse 0.033621
motorbike 0.000000
sheep 0.006824
train 0.065891
elephant 0.025006
bear 0.011893
zebra 0.084688
giraffe 0.031950
truck 0.017976
person 0.000000
```

As observed with the evaluation metrics, the model still lacks detection capabilities on both known and unknown objects due to the **issue regarding box localization**.

## 5. Library &Open source

### Reference codebases:

- OW-DETR:https://github.com/akshitac8/OW-DETR
- FOMO: https://github.com/orrzohar/FOMO
- Imported packages: OwlViTProcessor, OwlViTForObjectDetection, OwlViTConfig, OwlViTModel,OWEvaluator
- Major implementations are in files: main.py, engine.py, and OW-FOMO.py,

## 6. Contribution

| Member | Contribution |
|---|---|
| 312511814 Tom Broudin | **OWOD part of proposal**, method design |
| 312553002 蘇煒閎 | Method part of proposal, **coding, debugging, experiments**, parts of final report, method design |
| 312605013 張家倫 | OVD part of proposal, **demo presentation, final report summary**, method design |
| 312605024 戴麒恩 | CLIP part of proposal, **code comments**, **parts of pseudo code and coding**, method design |