Université de Montréal

« Where is my Stuff » An Indoor Object Detection Application For Visually Impaired Users

par

Ji Zhou Wang

Département d'informatique et de recherche opérationnelle (DIRO)
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique, Apprentissage automatique

January 5, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

« Where is my Stuff » An Indoor Object Detection Application For Visually Impaired Users

présenté par

Ji Zhou Wang

a été évalué par un jury composé des personnes suivantes :

Nom du président du jury
(président-rapporteur)

Nom du directeur de recherche
(directeur de recherche)

Nom du membre de jury
(membre du jury)

Student and Internship details

Student Name	Ji Zhou Wang
Student ID	20183404
Student Email	jizhou.wang@umontreal.ca
Internship Project	"Where is my stuff"
Company Name	Humanware Technologies
Company Website	`https://www.humanware.com/fr-canada/home`
Internship Supervisor	Francois Boutrouille
Internship Supervisor Email	francois.boutrouille@humanware.com

Abstract

This project focuses on helping visually impaired people detect common indoor objects using deep learning object detection models. The primary objective is to train a model that is robust to out-of-distribution class samples, meeting a precision and recall score threshold for each class and to provide feedback and guidance to the user through natural language on a portable device.

With the advent of large pretrained Foundation models for image recognition tasks such as CLIP [**33**] and ALIGN [**19**] (trained on millions of image/text pairs) in detecting out-of-distribution (OOD) samples, research has been further focused on utilizing these models for object detection. In this project, we explored foundation models such as ViLD [**15**] and RegionCLIP [**45**].

By using different methods of data augmentation and fine-tuning with different hyper-parameters for these models, we were able to quantitatively and qualitatively analyze the result of the detection proposed by evaluating the mAPs of each class. Precision-recall curves were graphed giving insight into different score threshold values meeting our criteria and visualizing the regions proposed by the network using a t-SNE 3D plot allowed us to distinguish false positive samples for further finetuning. The performance of foundation object detection models is then compared to more classical object detection models such as EfficientDet [**39**] showing improvement when inference on an OOD dataset.

Keywords: Object detection, Foundation models, CLIP, ViLD, RegionCLIP, OOD, mAP.

Contents

List of tables

List of figures

List of acronyms and abbreviations

OOD	Out-of-Distribution, a data setting where test data distribution is different from the train data distribution.
AP	Average Precision or mAP for mean Average Precision, a metric to measure the accuracy of object detectors.
SSRL	Self-Supervised Representation Learning, a deep learning method that learns from unlabeled data, provides a generalized feature representation.
RoI	Region of Interest, the bounding box locations of the detected object on the image.
RGB	Red, Green, and Blue color channels in an image.
RPN	Region Proposal Network, A vision backbone that proposes bounding box locations of objects in an image and ranks them by their objectness score.
CNN	Convolutional Neural Network, a neural network that analyzes visual imagery with shift-invariance.

IoU	Intersection over Union, an evaluation metric used to measure the accuracy of an object detector. The area where each bounding box overlap over the area of their union.
VQA	Visual Question Answering, dataset containing open-ended questions about images.
NMS	Non Maximum Suppression, a technique to suppress multiple bounding boxes over lapping the same object and obtaining the most relevant one.
LSJ	Large Scale Jitter, a data augmentation technique used in copy-paste [**12**] augmentation. It randomly scales and crops the input image.
LP	Linear Probe, a single (linear) layer that is added on top the last output feature layer of a model.
GT	Ground Truth, the true class label and location of the bounding box in the input image.

Acknowledgements

I'm grateful for having Francois Boutrouille, Ahmad Chamseddine, and Alain Bélanger as my colleague and supervisors during my time at Humanware. As well as with the support of University of Montreal and Mila - Quebec AI institute from my mentor, Mandana Samiei and advisor, Chloe Lacelle. They all have given me support and direction in terms of project planning, problem-solving, and resource management to help me achieve my prospects during the Internship.

Introduction

Founded in 1988, Humanware Technologies is a company that develops technologies for the visually impaired community. By providing technologies that adapt to the needs of their users from navigation, and magnification to Braille reading, Humanware's mission is to empower people with visual impairment to extend their degrees of freedom. With the recent advancements in AI and deep learning models, Humanware is consistently in search of innovation and integrating tools to provide an optimal experience for its users.

Problem background and Approach

One of the projects called "Where is my stuff" requires the localization of several indoor/in-building objects that are pertinent to the use case for a visually impaired user. As the question imposed by "Where is my stuff?" concerns the most frequently searched objects, the current list of categories includes 5 classes which are Elevator Doors, Trash Can, Wallet, Keychain, and Wall Outlet. The primary goal is to train an object detection model to locate these objects of interest and later be able to guide the user to their location with voice assistance. Two examples of indoor scenes are shown below. The dataset consists of scenes with close-up images of each individual class and distant images of multiple class objects at a higher scene variance.

Object detection based on deep learning models has been experimented with previously by Humanware such as EfficientDet [**39**]. Using data augmentation and model fine-tuning, a baseline performance was achieved. Although due to the variability of the object in different environments and settings (eg. continental/cultural differences), the OOD robustness of the model could be improved further. As it follows, foundation models were the next area of interest for research as they are highly capable of "zero-shot learning" and OOD robustness in transfer learning.

Foundation models in object detection

Models that are trained on a *large* quantity of unlabeled data through self-supervised representation learning (SSRL) [**10**] are called **Foundation Models**. These models can be

(a) Scene 1.

(b) Scene 2.

Two indoor scene examples.

used for many downstream tasks depending on the modality these models are trained on. As per CLIP [**33**] and ALIGN [**19**] trained on image/text pairs, these models can classify categories for any visual classification tasks, even for tasks it was not previously trained on (hence zero-shot capabilities).

From these foundation models come their object detector implementations namely **ViLD** [**15**] and **RegionCLIP** [**45**].

These object detection models are trained contrastively by aligning image regions and object concept's word embeddings obtained from the language model trained on a base foundation model such as CLIP. Using the image encoder from the foundation model (CLIP), a student network is trained from the proposed image regions with their RoI features aligned with the teacher network (CLIP) through knowledge distillation. A key difference between RegionCLIP and ViLD is that ViLD distills the regional proposals after feeding the region proposal to a pretrained CLIP model while RegionCLIP distills the region proposals after feeding the whole image through a pretrained CLIP model.

Furthermore, in research, multimodal foundation models such as Florence [**43**] and Flamingo [**1**], which leverage the range of representation to expand the representations from coarse (scene) to fine (object), from static (images) to dynamic (videos), and from RGB to multiple modalities (caption, depth). By incorporating a universal visual-language representations from Web-scale image-text data, these models can be easily adapted for various computer vision tasks, such as classification, object detection, VQA, image caption, video retrieval, and action recognition.

This project's focus will be to describe the results from experiments with ViLD and RegionCLIP as their implementations were open-sourced on GitHub (RegionCLIP, ViLD). Extensive fine-tuning was conducted with RegionCLIP as there was more familiarity with the environment (Detectron2, PyTorch).

Chapter 1

Previous work and literature review

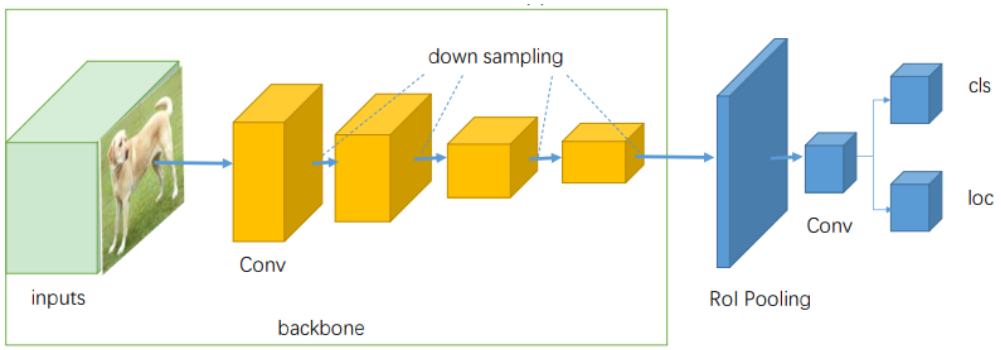
Before arriving at Humanware, indoor object detection has been previously experimented on with state-of-the-art model (in 2021) EfficientDet [**39**] and Yolov5 [**20**]. These object detectors weren't completely able to adapt to *real-life* situations such as the detection of small objects at medium to high distances (objects occupying less than 200px in area in the image) and classes with high false positives such as "Trash can". From these challenges, we would like to understand and utilize the latest advancement in foundation models for object detection to tackle these problems and potentially achieve a sufficient precision and recall score threshold.

1.1. Classical Object Detection Models

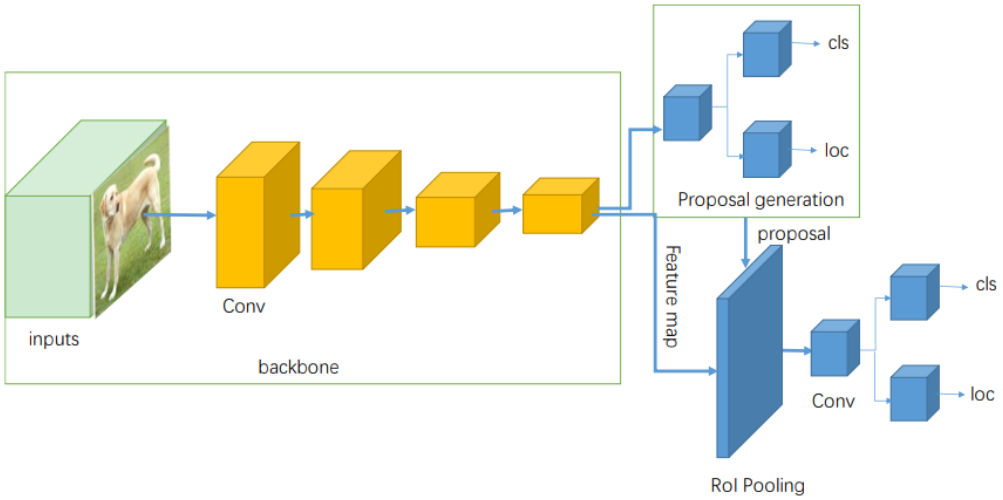
Advancements in object detection have been at the forefront of computer vision in the last decade with applications such as autonomous driving, facial recognition, hand gesture recognition, etc. Different models were developed with their unique pipelines and algorithms from one-stage detectors such as Yolo [**35**], EfficientDet [**39**], RetinaNet [**24**] and SSD [**27**] to two-stage detectors such as Faster R-CNN [**36**] and with FPN networks [**23**] shown in figure 1.3 where each feature level generates an object class prediction. Two-stage detectors utilize an RPN, shown in figure 1(b), to approximate the object regions from the image features passed through the CNN (Conv) before feeding these regions into another network for classification and bounding box regression. This process can be time-consuming as there are many inference steps per image. On the other hand, one-stage detectors, shown in figure 1(a), predict bounding boxes without the RPN thus allowing faster inference, though the trade-off may be reduced detection accuracy.

1.1.1. Yolo

The first Yolo [**35**] model is a one-stage detector that applies a single neural network to the image and divides it into SxS grids while predicting bounding boxes' objectness confidence



(a) One-stage Detector



(b) Two-stage Detector

Fig. 1.1. General object detector Architectures.

score, box offsets, and class probabilities for each grid all at once. Each grid has a predefined number of anchor boxes, these boxes are weighted by the predicted probabilities of each class. If the center of an object falls into a region then it becomes "responsible" for detecting that object shown in figure 1.2. The boxes are then selected based on confidence and IoU threshold scores.

Every iteration of Yolo incrementally improves inference speed and classification performance through the change of image encoder backbone architecture. Before the start of the internship, Yolov5 [**20**] was the latest model trained and tested by Humanware on their "Autonomous Blind Pedestrian" project used for house number recognition and crosswalk detection. It includes many new functionalities described on GitHub such as model ensembling, hyperparameter evolution, etc.

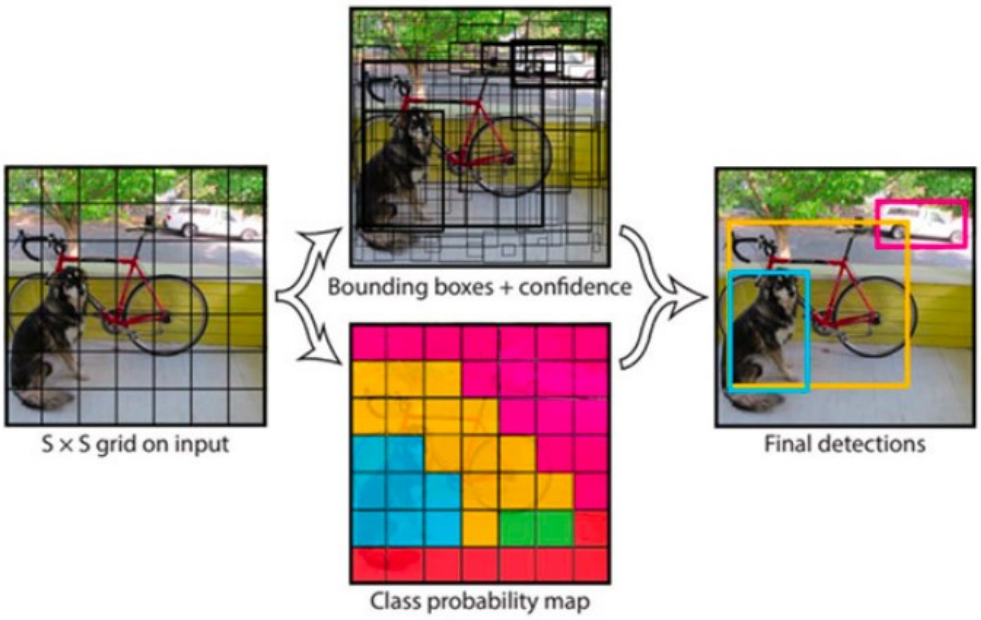


Fig. 1.2

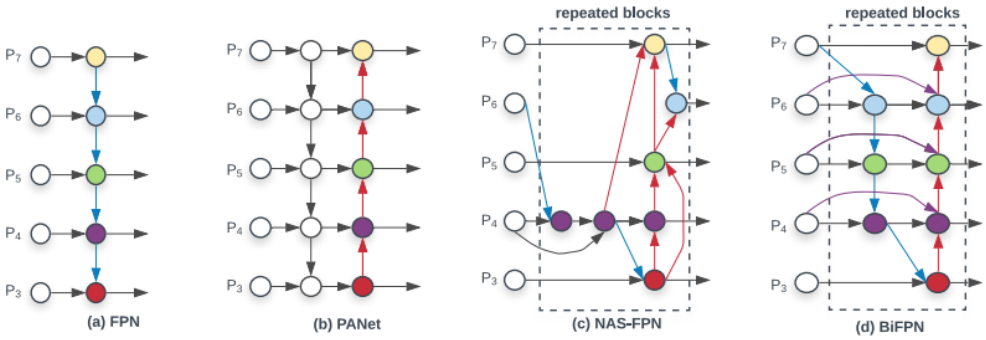


Fig. 1.3. Feature network designs

1.1.2. EfficientDet

Based on the backbone network implementation of EfficientNet [**38**] comes EfficientDet [**39**]. Like EfficientNet, EfficientDet's image encoder backbone compound scales the resolution, depth and width for all feature networks. In addition, it also scales the box/class prediction networks at the same time. The model architecture also incorporates a weighted bi-directional FPN (BiFPN) which allows the network to synergize with the scaling shown in figure 1.3. BiFPN utilizes ideas from FPN [**23**], PANet [**26**] and NAS-FPN [**13**] allowing information to flow from both top-down and bottom-up directions where features at different resolutions can have variable contributions to the output features of the network. It also optimizes the cross-scale connections by adding skip connections from the original to the output node.

Table 1.1. EfficientDet AP scores evaluated on respective datasets

Dataset	mAP	AP50	Elevator doors	Keychain	Wallet	Trash can	Wall oulet
Awakening	0.73	0.92	0.78	0.63	0.70	0.88	0.78
Humanware	0.65	0.81	0.81	0.55	0.67	0.84	0.49

Experiments were conducted with EfficientDet-D2 The results with the final model trained on the Awakening Vector dataset are shown in table 1.1 with 65% mAP on the Humanware Collected dataset and 73% mAP on the Awakening Vector dataset.

1.2. Foundation Object Detection Models

Foundation models are constructed with the training of a large quantity of unlabeled data through SSRL. GPT-3 [**4**] was one of the first foundation models that found success in language tasks by incrementally increasing the size of the training data. This phenomenon called Neural Scaling Laws [**21**] shows that loss scales as a power law with the model, dataset, and compute sizes. It was able to generalize to many language subtasks with few-shot learning capabilities such as translation, question-answering, etc. Similarly, for computer vision models, CLIP [**33**], ALIGN [**19**] and DALL-e [**34**] are foundation models pretrained on image and text pairs utilized on downstream tasks such as image classification, captioning, and generation. With these new foundation image encoder models, one can employ the backbone of these networks in combination with object detection networks to create a foundation object detection model; shown in ViLD [**15**] and RegionCLIP [**45**] which both incorporated CLIP into their models through student-teacher knowledge distillation. It has shown superior performance in open-vocabulary object detection tasks (object categories not seen in training) inferring OOD robustness.

1.2.1. CLIP

CLIP [**33**] is a foundation model pretrained on 400 million image and text pairs from Common Crawl. Through contrastive loss by minimizing the cosine similarity between the text and image encoders in a multi-modal space, CLIP is able to pretrain both encoders shown in figure 1.4. It's pretrained with GPT-2 [**4**] as text encoder and many variants of ResNet [**18**] and ViT [**8**] as image encoders. ViT has the distinct advantage of the Transformers architecture, which allows for self-attention weights in the model and has shown to be 3x more compute efficient in CLIP compared to ResNet.

With text prompts, CLIP can be used to reference learned visual concepts that may be present in the input image, this enables zero-shot transfer shown in figure 1.5. Custom-labelled datasets can be finetuned by adding a linear probe or LP to classify the image through the image encoder. Pretrained weights of the LP are defined by the text encoder with

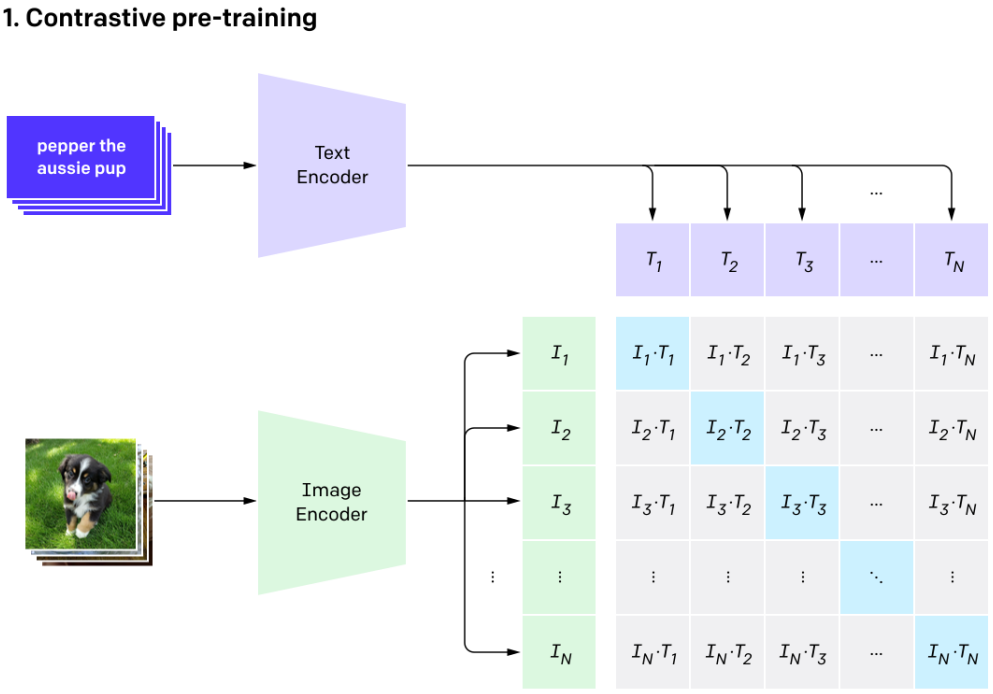


Fig. 1.4

the label text passed through the text encoder as text embeddings. These text embeddings can be ensembled through prompt engineering to better encode the description of the image in finetuning and inference.

Overall the experiments from CLIP have shown that its zero-shot performance is robust to distribution and domain shift. Its failure modes include inference for abstract and numerical tasks such as object spacial relation and counting and having ill-defined text prompts. The model's performance for the in-domain task can be improved after finetuning with a linear probe, though at the cost of OOD robustness.

Experiments were previously conducted by Humanware using an LP as an intermediary between image classification and object detection to measure the performance of classifying "high-variance" image samples with multi-label classification. A caveat of this method is that it only detects the object's presence in the image but does not exactly locate the object. The results seemed promising, above 97% on the Awakening Vector dataset and 77% on the Humanware Collected dataset with both ResNet [**18**] and ViT [**8**] CLIP image encoders.

1.2.2. ViLD

Vision and **L**anguage Knowledge **D**istillation or ViLD [**15**] is a newly proposed Foundational Object Detection Model which aims at improving the performance of detecting novel (OOD) classes through a pretrained image/text encoder and a two-stage detector. The network is trained through student-teacher knowledge distillation that aligns the pretrained

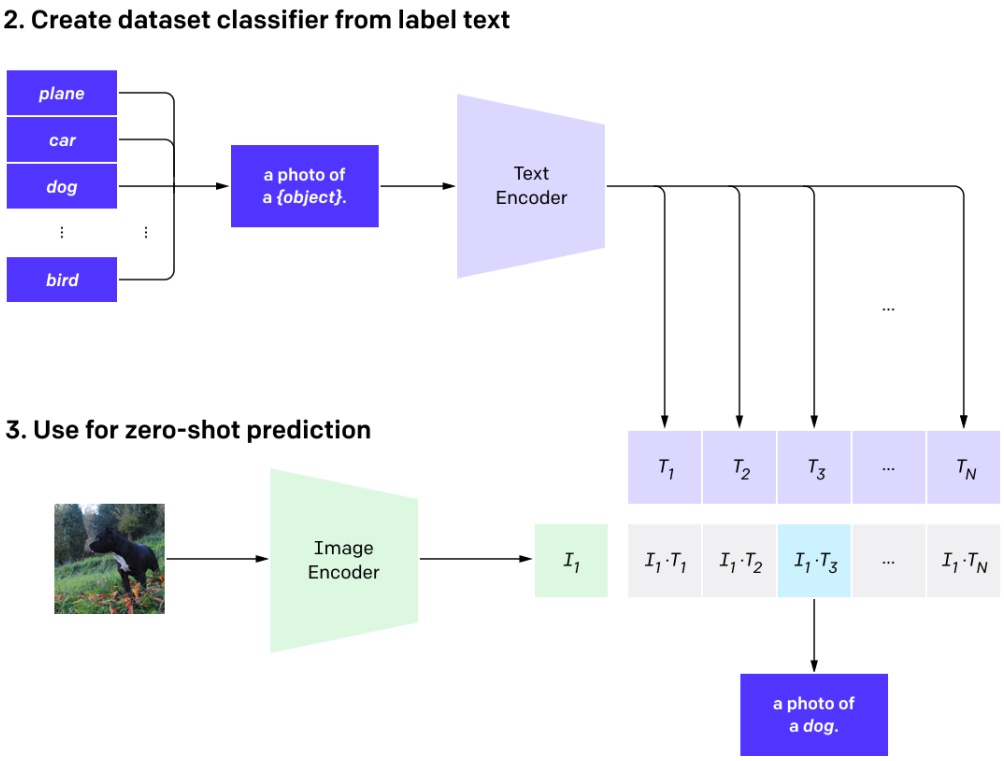


Fig. 1.5

image encoder (teacher) and the region embeddings of the Mask R-CNN [**17**] (student). The teacher region embeddings are obtained by feeding the region proposals generated by the RPN into the pretrained image encoder. The label text embeddings are computed through the pretrained text encoder with prompt engineering. Finally, by minimizing the cosine similarity between the region embedding and the label embeddings through contrastive cross entropy loss, the model is trained to for open-vocabulary detection (transferable to other datasets). The whole pipeline is shown in figure 1.6.

ViLD consists of 3 main models, ViLD-text, ViLD-image and ViLD (Combined ViLD-text and ViLD-image) shown in figure 1.7.

ViLD-text obtains the text embeddings by feeding the label names with a normalized ensemble of prompt templates, hence prompt engineering such as "a photo of label in the scene", into the pretrained text encoder. These generated text embeddings $\mathbf{t}_{|C_B|}$ are compared with each region embeddings $\mathbf{e}_r$ through cosine similarity before applying a softmax activation with temperature τ to compute the cross entropy loss shown in equation 1.2.1. Note y_r denotes the class label of the region r

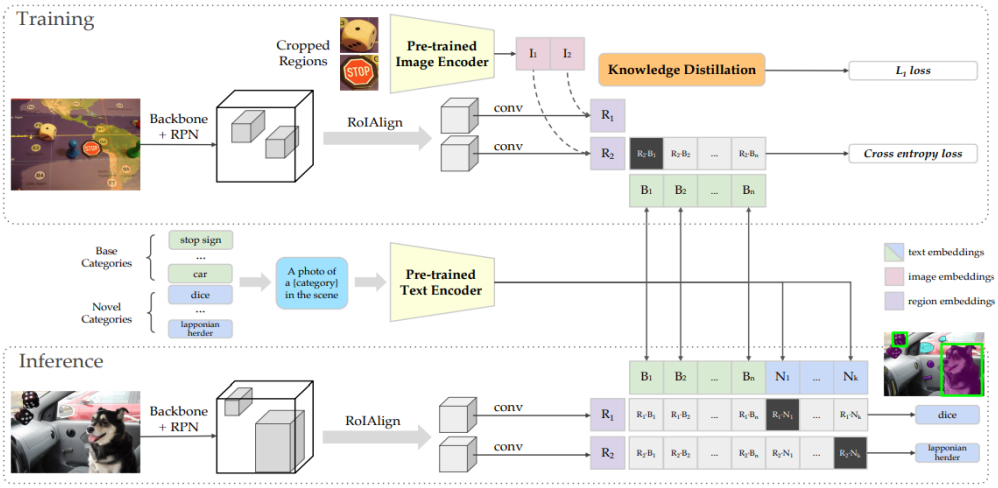


Fig. 1.6. ViLD training and inference pipeline

$$
\begin{aligned}
\mathbf{e}_r &= \mathcal{R}(\phi(I), r) \\
\mathbf{z}(r) &= \left[sim\left(\mathbf{e}_r, \mathbf{e}_{bg}\right), sim\left(\mathbf{e}_r, \mathbf{t}_1\right), \cdots, sim\left(\mathbf{e}_r, \mathbf{t}_{|C_B|}\right) \right] \\
\mathcal{L}_{\text{ViLD-text}} &= \frac{1}{N} \sum_{r \in P} \mathcal{L}_{\text{CE}}\left(softmax(\mathbf{z}(r)/\tau), y_r\right)
\end{aligned}
\tag{1.2.1}
$$

ViLD-image obtains the image embeddings (normalized ensemble of 1× and 1.5× crops) from the region proposals generated by the pretrained image encoder $\mathcal{R}$. Through knowledge distillation, a Mask R-CNN is trained to align these region embeddings. Note that the RPN is trained on the base (frequent and common) categories of the LVIS [**16**] dataset and is still able to propose regions with novel categories from LVIS. Unlike ViLD-text, ViLD-image distills knowledge from all categories (base and novel). A $\mathcal{L}_1$ loss is applied between the region and image embeddings shown in equation 1.2.2.

$$
\mathcal{L}_{\text{ViLD-image}} = \frac{1}{M} \sum_{\tilde{r} \in \tilde{P}} \left\| \mathcal{V}(\text{crop}(I, \tilde{r}_{\{1\times, 1.5times\}})) - \mathcal{R}(\phi(I), \tilde{r}) \right\| \tag{1.2.2}
$$

The final ViLD model combines both ViLD-text and ViLD-image as a weighted sum shown in equation 1.2.3 with w, a hyperparameter set to 0.5

$$
\mathcal{L}_{\text{ViLD}} = \mathcal{L}_{\text{ViLD-text}} + w * \mathcal{L}_{\text{ViLD-image}} \tag{1.2.3}
$$

Benchmarked on LVIS [**16**] dataset, ViLD was able to achieve 16.7 AP on novel classes (337 categories) with ResNet-50-FPN [**18**] as the backbone and 19.8 AP on novel classes with ResNet-152-FPN as the backbone. On COCO [**25**] dataset, ViLD was able to achieve 27.6 AP on novel classes and 59.5 AP on base classes.

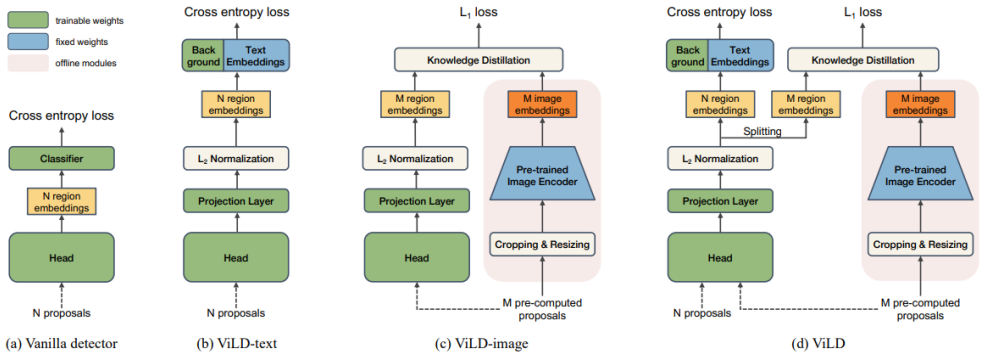


Fig. 1.7. ViLD model architectures

1.2.3. RegionCLIP

Very similar to ViLD, RegionCLIP [**45**] also aims to detect OOD classes through a pretrained image/text encoder. It also aligns the image region embeddings obtained from the RPN (R-CNN with RoIAlign [**17**]) through knowledge distillation. Shown in figure 1.8, box 1 summarizes how CLIP aligns the image-text pairs. Box 2 shows RegionCLIP's aligning of region-text pairs, with region embedding distilled from CLIP and a concept pool of text embeddings generated from prompt engineering passed through CLIP's text encoder. The pretrained encoder is able to generate the region-text pairs for contrastive learning. Lastly, in box 3 after pretraining on the regions, the object detector is able to transfer detect to a dataset with novel categories.

A few differences between RegionCLIP and ViLD is that RegionCLIP utilizes KL-divergence $\mathcal{L}_{KL}$ for its image-level distillation loss as oppose to $\mathcal{L}_1$ loss in ViLD. RegionCLIP also used a separate dataset, Conceptual Caption data (CC3M) for pretraining the image-text pairs while filtering object labels whose frequency is lower than 100 resulting in 6790 categories, while ViLD uses the default LVIS base categories for pretraining (866 categories). During transfer learning, RegionCLIP is trained without using Copy-Paste [**12**] data augmentation, reducing training time by 16x compared to ViLD. In addition, RegionCLIP also applies focal scaling [**24**] to the base categories of the new dataset to alleviate the forgetting of previously learned object concepts in pretraining. This is very pertinent to Humanware's case as there are only 5 categories, and we would like to have a model that is robust to distributional changes in concepts similar to these 5 categories while not overfitting to the training samples during transfer learning.

The AP results of RegionCLIP experiments have shown to improve over ViLD with on LVIS [**16**] dataset, RegionCLIP was able to achieve 17.1 AP on novel classes (337 categories) with ResNet-50-C4 [**18**] as the backbone and 22.0 AP on novel classes with ResNet-50x4-C4 as the backbone. On COCO [**25**] dataset, RegionCLIP was able to achieve 39.3 AP on novel

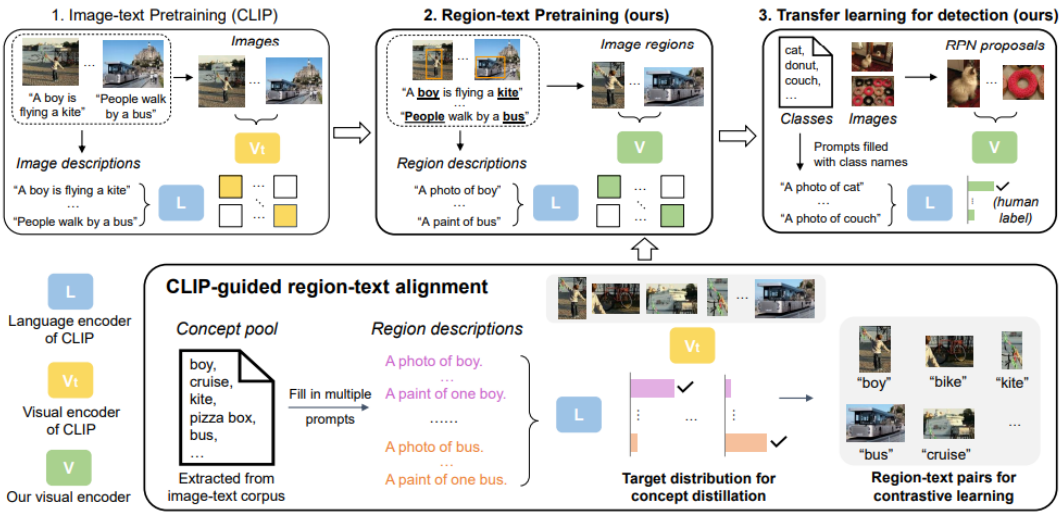


Fig. 1.8. RegionCLIP model pipeline with Comparison to CLIP

Table 1.2. AP percentages comparison on COCO and LVIS datasets for ViLD and RegionCLIP

Model		COCO			LVIS			
Architechture	Backbone	Novel	Base	All	APr	APc	APf	mAP
ViLD	RN50-FPN	27.6	59.5	51.3	16.7	26.5	34.2	27.8
RegionCLIP	RN50-C4	31.4	57.1	50.4	17.1	27.4	34.0	28.2
ViLD	RN152-FPN	-	-	-	19.8	27.1	34.5	28.7
RegionCLIP	RN50x4-C4	39.3	61.6	55.7	22.0	32.1	36.9	32.3

classes and 61.6 AP on base classes. A more detailed comparison of the results is shown in table 1.2.

In this report, our main focus will be on experimentation with RegionCLIP as its code was readily available on GitHub with pretrained model checkpoints, zero-shot inference, transfer learning and region feature extraction described in the documentation.

1.3. Other Foundation Models

Florence [**43**] is a multimodal computer vision foundation model that captures a shared image-label-text description representation using a unified image-text contrastive learning objective (UniCL) [**42**] Its image encoder backbone is a modified Swin Transformer [**28**] with features extracted through global average pooling. The model incorporates many sophisticated adaptation models such as Dynamic Head Adaptor [**7**] for object-level representation and METER Adaptor [**9**] for fine-grained representation which can support many downstream tasks such as object detection, VQA, image caption, video retrieval, and action recognition. An overview of the model is shown in figure 1.9.

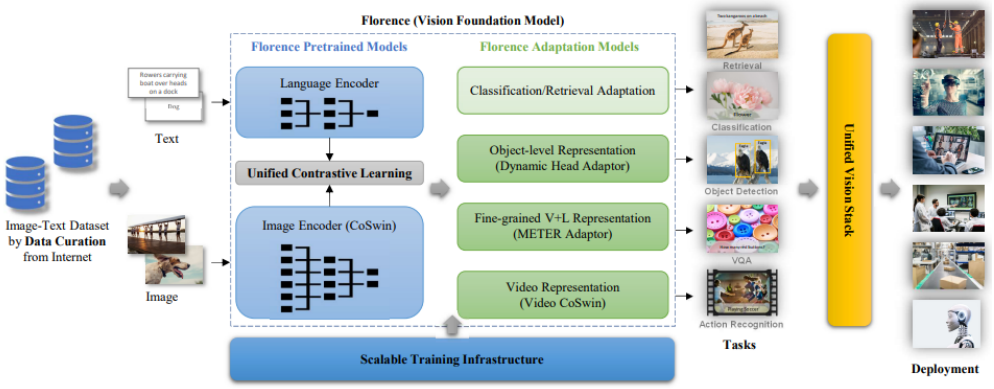


Fig. 1.9. Florence model workflow and pipeline

Recently, another family of visual language models called Flamingo [**1**] tries to bridge powerful pretrained vision-only and language-only foundation models into a single multimodal model. These models are visually-conditioned autoregressive text generation models and are able to seamlessly handle sequences of visual and textual data such as interleaved text, image and video inputs and produce text as output. It leverages two pretrained frozen vision and language models through Perceiver Resampler and GATED XATTN respectively to connect them in a way that preserves the knowledge they gain through pretraining described further in the paper. This allows for in-context few-shot learning by prompting the model with task-specific examples such as VQA. Flamingo was able to outperform models fine-tuned on task-specific data in many downstream tasks. An overview of the model is shown in figure 1.10

These models are considered for Humanware's future research as they hold greater adaptability in terms of the "Where is my stuff?" project as the model may not need to detect the actual object and can seamlessly verbalize the location of the object if it is present in the image through VQA.

1.4. Object Detection Metrics

It is important to define a quantitative measure in object detection where the classification accuracy and the location of the detected bounding box relative to the ground truth (GT) is taken into account.

Average Precision or AP is one of the top metrics that measures the average precision value for recall value from ranged from 0 to 1. To understand AP, one must understand precision and recall. Given a certain classification score threshold, **Precision** in object detection defines the number of relevant objects detected out of all detected objects and **Recall** in object detection defines the number of relevant objects detected out of all GT objects shown in figure 1.11. A precision-recall curve can then be generated to show the

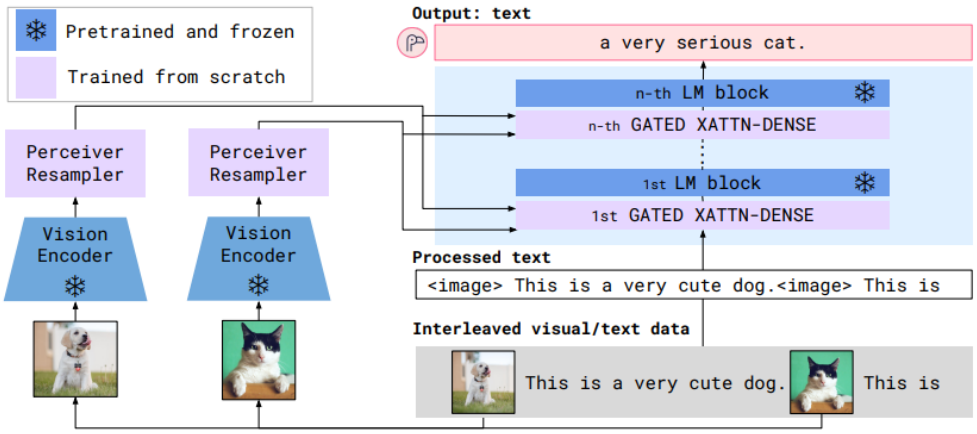


Fig. 1.10. Flamingo architecture overview

trade-off between precision and recall for different score thresholds with an example shown in figure 1.12. Finally, the Average Precision is shown as the area under the curve calculated as a weighted sum of precisions at each threshold where the weight is the increase in recall.

We note that for an AP value, one must define the IoU of a detection overlapping the GT bounding box for the detected bounding box to be labelled as a true positive otherwise a false positive.

For the purpose of the experiments conducted at Humanware, AP is referred to as AP50 or AP with detection IoU set to 0.5 and mAP is referred to as the mean of all APs from AP50 to AP95 with an interval of 0.05. The priority for this project is to aim for a recall rate of 60% at a precision of 80% to 90% depending on the object class by reducing false positive predictions through methods such as negative sample training, focal scaling [**24**] etc.

In terms of model bounding box detections, NMS is used to filter many detected bounding boxes to obtain the most relevant one based on criteria such as IoU and classification score. There are two types of NMS, Hard NMS and Soft NMS [**3**]. Hard NMS first selects the proposal with the highest score, these selected proposals are compared with each other by their IoU and if the other proposal's IoU is greater than a set NMS threshold then remove that proposal. Looping through all the proposals ordered by confidence scores will generate a list of final proposals. Soft NMS instead of removing the proposals with high IoU, uses a gaussian function that decays the detection scores of all other objects proportional to the IoU value and keeps the proposals below the NMS threshold.

For our experiments, we have used Hard NMS for model inference.

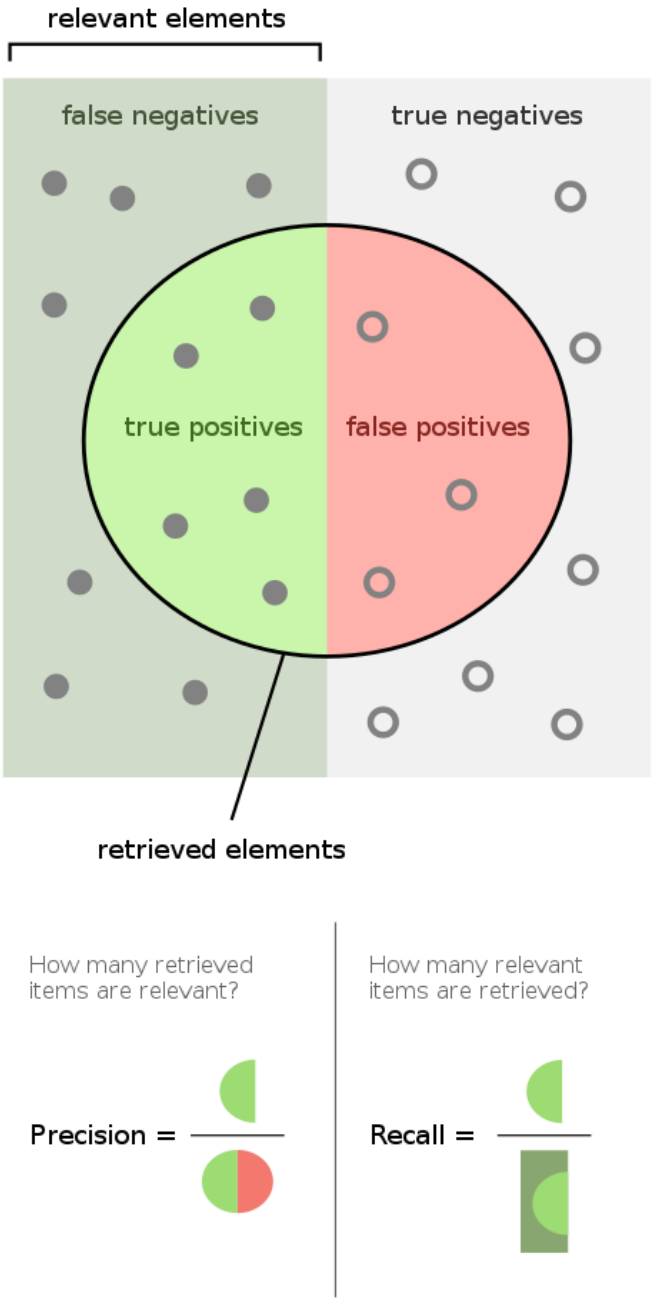


Fig. 1.11. Precision and Recall

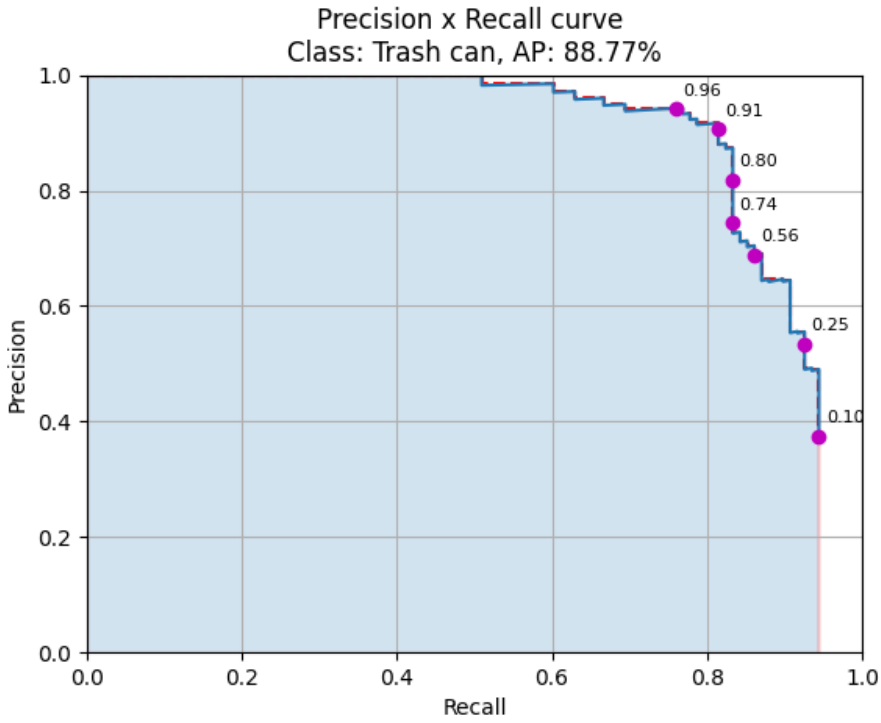


Fig. 1.12. PR Curve on "Trash can" class with purple dots showing score thresholds.

Chapter 2

Datasets

Many dataset experiments were conducted with two private datasets companies, Awakening Vector (Awakening for short) and BasicAI, A "Combined" dataset with Awakening Vector and BasicAI was also generated for finetuning during training. Humanware employees have also collected their own crowd-sourced "real-life scenarios" photos as one of the test sets that aims to measure OOD generalization performance. Most of the manual labeling was done on labelme an image annotation tool. The main set of class labels used for detection were: Elevator doors, Keychain, Trash can, Wallet and Wall outlet. "Elevator button" was also included at the start of the internship but was discarded due to its difficulty in detection. In hindsight, if the model is able to detect an elevator door then it is reasonable to assume that the elevator button would be next to it.

Private datasets are split with 0.8/0.1/0.1 as train/validation/testing sets. To generate labels suitable for the input of training on the RegionCLIP model, one must convert the individual JSON label file's bounding box format from XYXY (Xmin, Ymin, Xmax, Ymax) to XYWH (Xmin, Ymin, Width, Height) and combine them all from the respective train/val/test set into a single JSON file.

Each individual class label condition should be well-defined prior to annotation, for example, imposing that key rings belong to a "Keychain" and labelling each individual key as a "Keychain". Annotation issues were reported to the company for relabeling. No more than 3 samples of the same objects in the dataset can be taken from different angles.

2.1. Humanware Collected

In total, there are 575 Humanware Collected (Humanware for short) images with 81 Trash cans, 197 Keychains, 115 Wallets, 339 Wall outlets, 11 Elevator buttons and 15 Elevator doors annotated. This dataset was only used as a testing set to measure OOD performance in relation to the samples trained on the private datasets which were mostly collected outside of Montreal, Canada. An example of an image is shown in figure 2.1, given labels

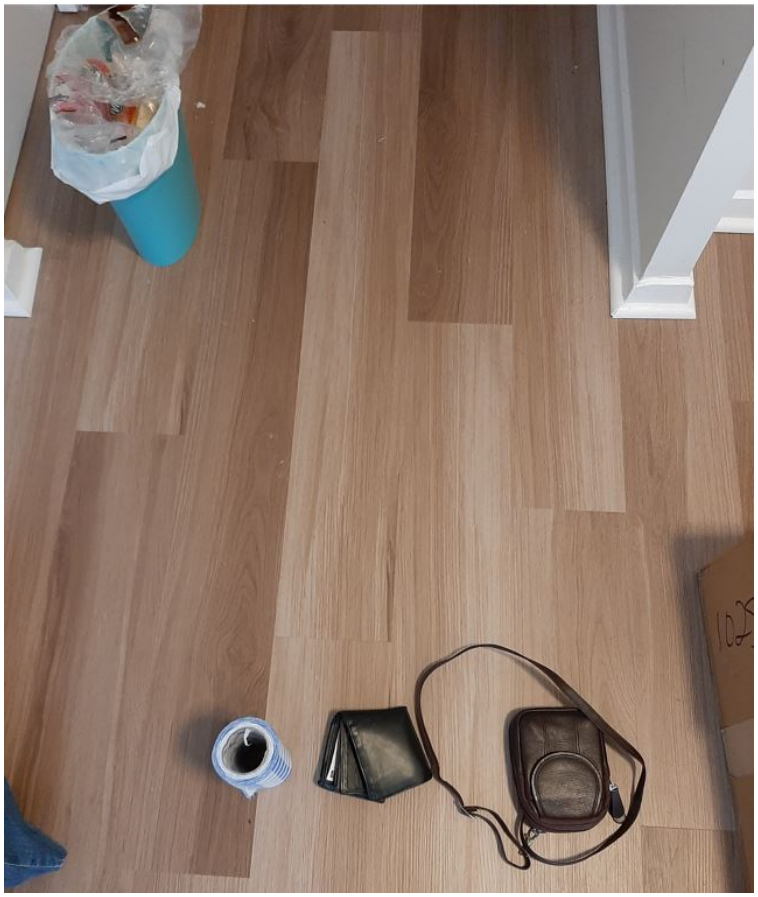


Fig. 2.1. A sample image (60.jpg) of Humanware Collected dataset

in JSON format as `{"imageWidth": 643, "imagePath": "60.jpg", "imageHeight": 764, "bbox": [{"label": "Trash can", "group_id": null, "x": 42, "y": 6, "width": 130, "height": 225}, {"label": "wallet", "group_id": null, "x": 260, "y": 594, "width": 110, "height": 100}]}`. From this JSON file, two classes, Trash can, and Wallet are labelled in the image.

We note the class imbalance in this dataset suggesting a future iteration of this dataset can incorporate more OOD samples of those classes.

2.2. Awakening Vector

For the Awakening Vector dataset (Awakening for short), there are a total of 29917 images with 6843 Trash cans, 6494 Keychains, 6147 Wallets, 7746 Wall outlets, 5850 Elevator buttons, and 8291 Elevator doors annotated. Most of these images are web-crawled using Baidu Search queries and are close-ups of each individual class. A sample of each class is shown in figure 2.2. Training on these images may finetune the model to be robust to ideal close-ups but in real-life scenarios where objects may be obscured, far or in different lighting may come as a challenge.

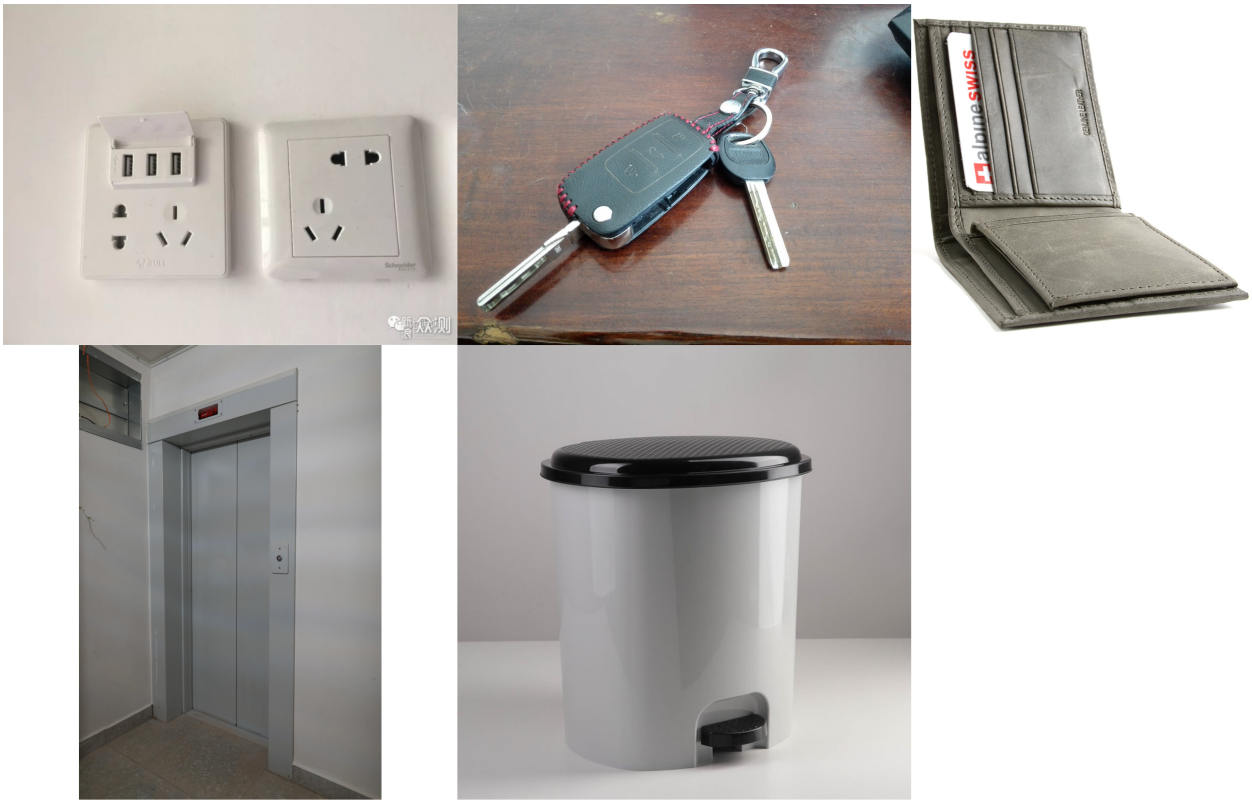


Fig. 2.2. 5 sample images of Awakening Vector dataset

2.3. BasicAI

To compensate for the lack of real-life scenario images in the Awakening dataset, the BasicAI dataset was acquired. There are a total of 4433 images with 1130 Trash cans, 1018 Keychains, 1031 Wallets, 1588 Wall outlets, and 1045 Elevator doors annotated. These images were captured (mostly from China) in different in-door scenes, and environments such as homes and offices. Many contain multiple class labels per image with samples shown in figure 2.3.

Later in the internship, we requested BasicAI to relabel the same dataset with an additional 63 classes. They are namely; door, container, window, frame, potted plant, armchair, backpack, vase, switch, chair, dining table, glass, water pot, curtain, shelf, basket, cabinet, bottle, stool, cup, bowl, drawer, pencil, remote, couch, bed, cell phone, kettle, closet, laptop, pan, food jar, tv, router, knob, monitor, lamp, book, lanyard, handle, keypad, keyboard, mouse, binder, plate, cable, refrigerator, wardrobe, speaker, power bar, hook, kitchen pot, oven, game console, bag, can, knife, sticker, label, vacuum, stove, coffee machine, and grinder. The number of samples for each of these class vary greatly (from as few as 1 to as many as 3000) as it is samples labeled from the original BasicAI dataset.

These curated classes are gathered after qualitatively analyzing the output of the trained models for false positives through T-SNE visualizations showing section 2.4.1. They are used

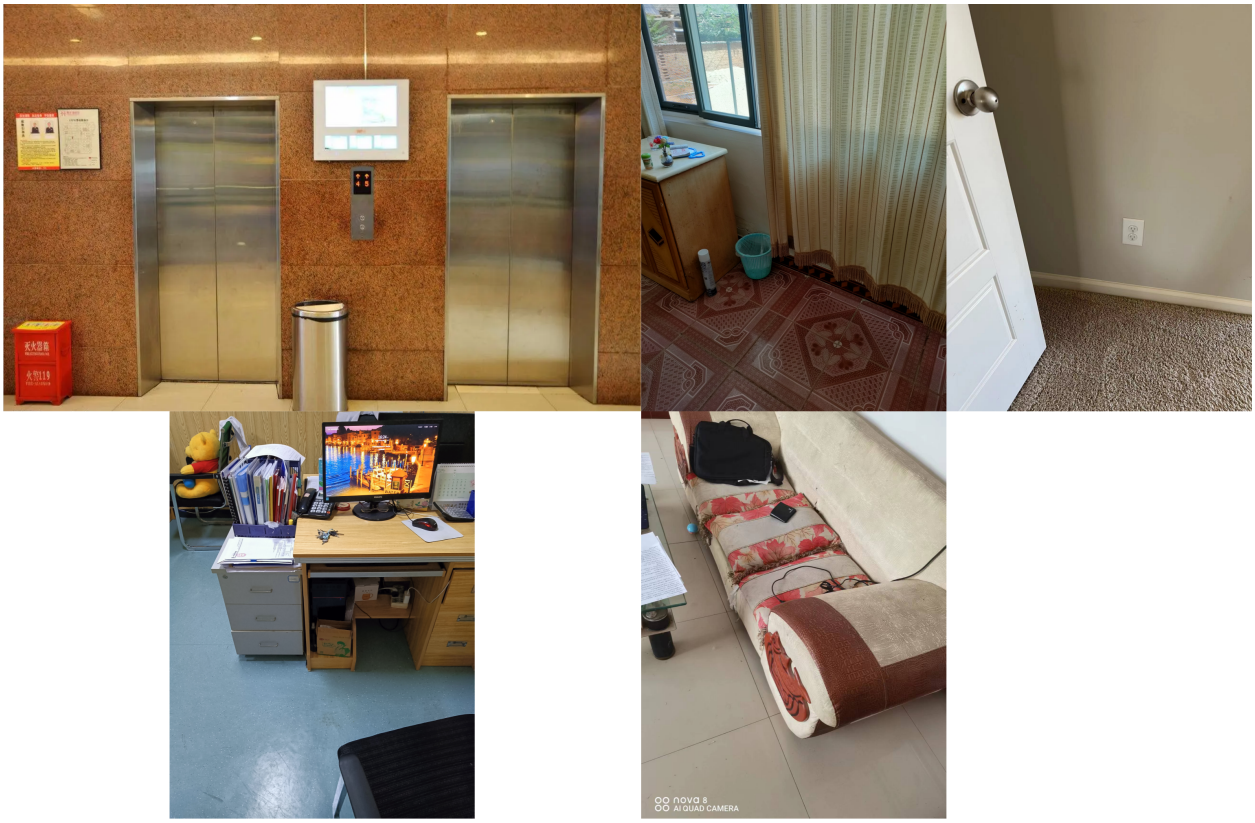


Fig. 2.3. 5 sample images of BasicAI dataset

to train the model to better separate out the false positive classes that are close in region-encoded feature space compared to the original classes. For example. some of the classes that are close to the "Trash can" class were "cup", "pot", "vase", "bowl", and "container".

2.3.1. COCO Indoor

Similarly, Humanware has decided to leverage the idea of training on other indoor classes by filtering out a list of 52628 images containing 48 Indoor objects from the COCO dataset. The list of 48 objects namely are; bottle, plate, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed mirror, dining table, window, desk, toilet, door, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, blender, book, clock, vase, scissors, teddy bear, hair drier, toothbrush, and hair brush. Some examples are shown in figure 2.4.

These images are then combined with the BasicAI dataset to create a new train and validation set for finetuning in hopes of better separating out false positive samples and improving AP in the 5 main classes.

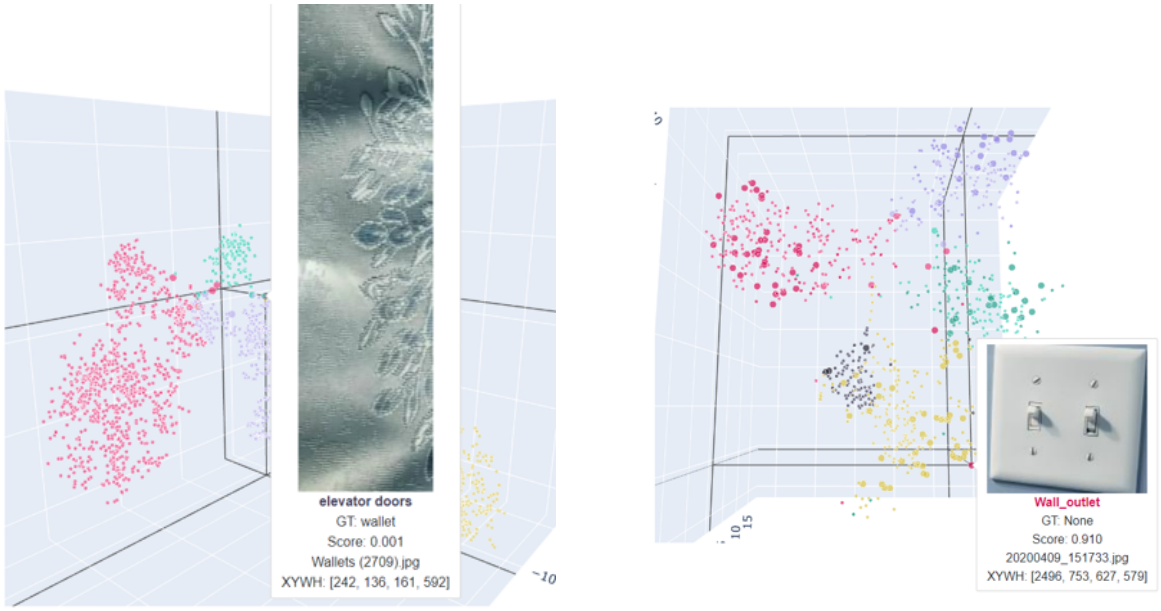
Fig. 2.4. 6 sample images of coco dataset

2.4. Image and Label Analysis

During the data preprocessing stage, we implemented a pipeline to check for image formats and convert them so that they are able to be accepted by the models that they're trained on as well as correcting the orientation of the image as they were sometimes rotated in relation to their JSON label file. Metadata in regards to the number of labels for each class and the mean area (in pixel) of these labels were printed with respect to their datasets. It was interesting to see that BasicAI dataset has a much smaller pixel area (20k) for "Keychain" and "Wall outlet" classes compared to Humanware and Awakening whose areas are at least 10x larger (200k). This is explained in more detail in the RPN section in regards to why it was difficult detecting those two classes in the BasicAI dataset.

2.4.1. T-SNE Visualizations

GT and RPN bounding boxes (After NMS) were also visualized after passing the regions through a finetuned model and obtaining their region feature representations before applying T-SNE in an interactive 3D plot. The visualizations were able to show us outlier labels but also inconsistent/incorrect ones with two examples shown in figure 2.5. This allows us to correct GT labels that are inconsistent, usually shown as an outlier in the visualization. It also helps us to qualitatively understand which class labels are more difficult to detect as



(a) Incorrect GT bounding box

(b) False positive RPN bounding box

Fig. 2.5. T-SNE 3D visualization of bounding boxes with 5 distinct colors showing 5 different classes.

shown by their false positives from the RPN bounding box detection outliers. Additional examples of the visual application are shown in appendix A.

Chapter 3

Methodology

The experiments were conducted on Humanware's dedicated server with Python and shell scripts. Two NVIDIA GeForce RTX 3090 each with 24GB of RAM were available for use. For RegionCLIP experiments, I was able to utilize multi-GPU Parallel processing for training and finetuning. For inference and to obtain region visual features, only a single GPU was needed. For ViLD experiments, the linear probe (LP) implementation was constructed on scikit-learn [**32**] without GPU processing.

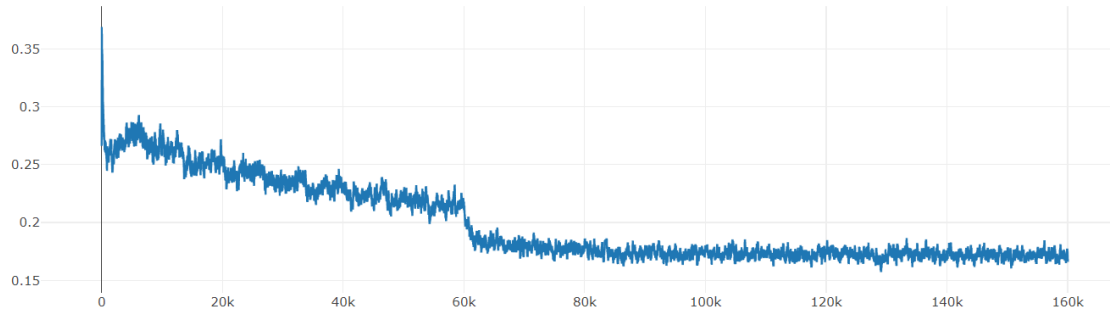
3.1. Experiment setup and Configs

Experiment tracking was set up for RegionCLIP on mlflow. It tracked hyperparameters such as epochs, batch size, data augmentations, learning rate parameters, and loss function parameters. It also tracked the AP of all 5 classes (Trash can, Wall outlet, elevator doors, keychain, wallet), the mAP, AP50, AP75, APl, APm, APs, and mAP of each class for every evaluation epoch on the validation set. The box regression, class, and total loss were also measured. An example of mAP and total loss graph is shown in figure 3.1.

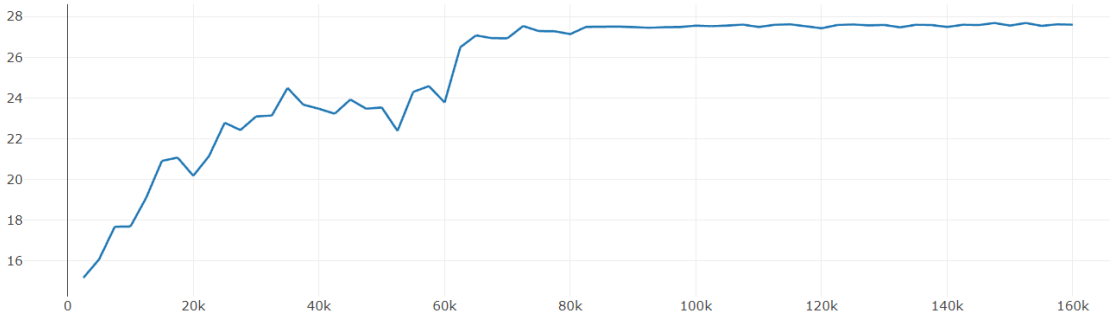
3.2. ViLD

At the start of the internship, ViLD [**15**] was the only model with code available. Although it only had outlined capabilities for zero-shot inference, therefore we decided on implementing a LP classification layer on top of the model region feature outputs from the last layer. The architectures proposed for the training of LP were decided to be Logistic Regression, XGBoost [**5**], and a soft ensemble model of the Logistic Regression, XGBoost and Zero-shot inference which sums the predicted probabilities for class labels and predicting the class label with the largest sum probability.

The configs was set with 0.5 IoU, 0.99 confidence score threshold for evaluating precision and recall.



(a) total loss graph



(b) mAP graph

Fig. 3.1. mlflow graphs

3.3. RegionCLIP

RegionCLIP [**45**] is a model built on top of Detectron2 [**41**], a platform for object detection written in PyTorch. It contains documentation related to model training, model checkpoints, finetuning, configs, and many other guides related to computer vision. We decided to take advantage of the documentation that RegionCLIP presented us with to finetune our own model with hyperparameter optimizations. It was convenient to hot swap different train/valid/testing sets for inference and we were able to customize its visualization script to generate RPN outputs before and after NMS filter as well as outputs showing the top # of predicted classes. RegionCLIP also outperforms ViLD in terms of object detection described in the paper therefore, we would like to minimize our resource expenditure and focus more on finetuning RegionCLIP.

In general, the settings were kept as default other than redefining the number of classes and their text embeddings for training and inference. The base learning rate is set at 0.002 and the number of iterations was set such that the AP improvements were levelling off shown in mlflow. The best model evaluated on the validation set using AP50 (the mean across all classes) metric was saved. Focal loss was enabled and different gamma and alpha settings were tested. In general, during training, focal loss helps with detecting classes that

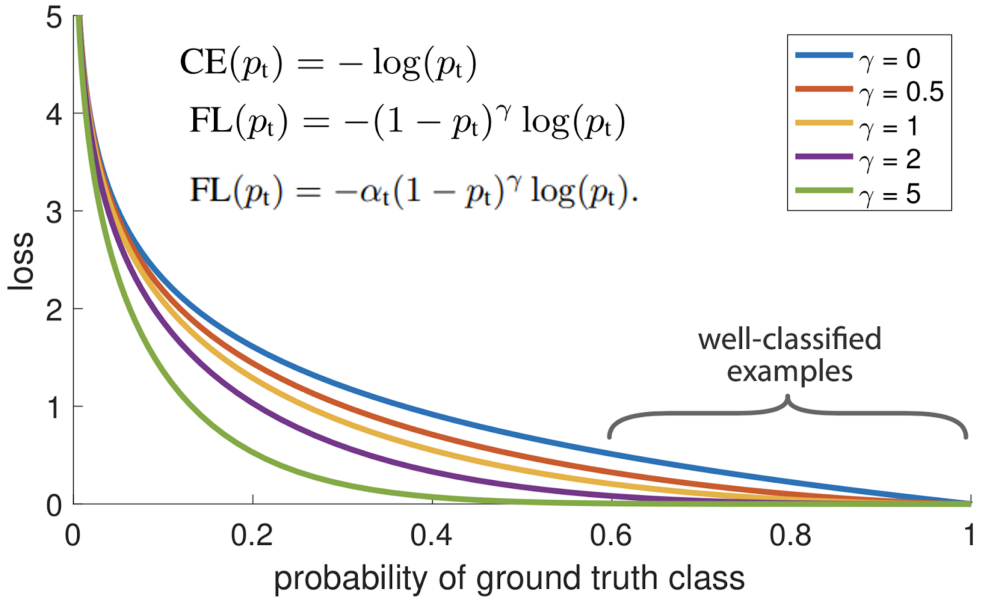


Fig. 3.2. Description of Focal loss (FL) with comparison to Cross Entropy (CE) loss. The images show the effect of changing the γ values and how it affects loss.

are imbalanced by assigning more weight to hard/misclassified examples and less weight to well-classified examples. Shown in figure 3.2, the higher the γ (Gamma), the more focus on difficult positive samples, and the smaller the α (Alpha), the more focus on negative samples. We also tried different optimizers such as Adam [**22**] and SGD. The batch size for each epoch needed to be set according to which ResNet (Batch size 8 for RN50 and Batch Size 2 for RN50x4) as it may overflow the GPU memory.

In terms of inference, in the classification head bounding box outputs, NMS was set to 0.6 with a confidence score threshold set to 0.001 (before visualization filters). After experimenting with different RPN settings, we have also decided to set the NMS at the RPN level to 0.7 and filter 12000 bounding boxes ranked by their objectness score pre-NMS and 2000 bounding boxes post-NMS. Given GT bounding boxes, the final inference evaluation outputs the mAP for all classes and each individual class.

3.3.1. Data augmentation

Every image given to RegionCLIP had to be augmented with `ResizeShortestEdge` which scales the shorter edge to a given size while keeping the same aspect ratio. The min size was 800 on the shorter edge and the max size was 1333 on the longer edge.

By default, the data augmentation pipeline for training was:

(1) ResizeShortestEdge: min size 800, max size 1333

(2) RandomFlip: horizontal flip

(3) (optional) RandomCrop: 0.9x of the image.

(4) (optional) LSJ: Scale the image between 0.1 and 2.0, pad/crop the image to 800x800 if needed.

(5) (optional) RandomRotation: choose between 90, 180, and 270 degrees.

CopyPaste [**12**] data augmentation was also considered but due to the resource constraint and the lack of segmentation labels, it was not used as the technique would be inconsistent with only bounding box labels shown by previous Humanware experiments.

Chapter 4

Experiment Results and Analysis

The experiment results for ViLD [**15**] and RegionCLIP are described in full detail below. The goal was to improve the AP performance on Humanware Collected and BasicAI datasets as they represent "real-life" scenarios with in-door scenes that are in context rather than Awakening Vector dataset.

4.1. ViLD

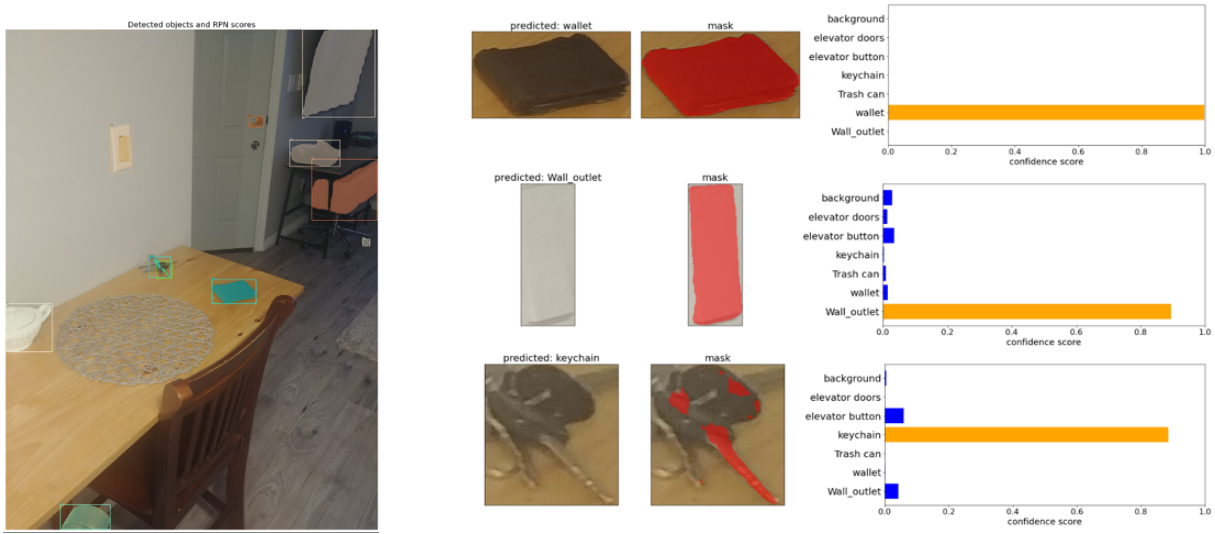
Given ViLD's region feature outputs, we conducted experiments described below using Awakening and Humanware datasets.

4.1.1. Zero-shot evaluation

First, we tried Zero-shot inference on the test sets, an example of the inference from the ViLD demo is shown in figure 4.1. The average precision and recall for all classes on the Awakening test set were 0.21 and 0.40 respectively. On the Humanware test set was 0.14 and 0.32 evaluated with configs described in section 3.

4.1.2. Linear Probe Comparisons

Training for ViLD LP was conducted on the Awakening dataset combined with a portion of the Humanware dataset as well. Another set of ViLD LP training was conducted without the Humanware dataset to evaluate the performance of each respective set. The results are shown in table 4.1 and in table 4.2. First, we note that CLIP's precision and recall values are only for class detection on the whole image and not localization within the image, therefore it is an easier task as shown by its high percentages. We can see from the results in table 4.1, ViLD zero-shot inference had a low precision-recall, especially on the Humanware dataset, the model wasn't always able to classify the classes even with a score threshold set at 0.99. Out of the three different models tried for LP, XGBoost had the highest on the Awakening (0.78) and Humanware (0.82) datasets with the model trained on both datasets.



(a) ViLD image output

(b) Some of the detected class and confidence scores.

Fig. 4.1. ViLD zero-shot inference output.

Base Model	Method	Awakening		Humanware	
		Precision	Recall	Precision	Recall
CLIP	Multiclass Logistic Regression	0.97	0.97	0.78	0.77
ViLD	Zero-shot	0.21	0.40	0.14	0.32
ViLD	Logistic Regression	0.77	0.48	0.67	0.37
ViLD	XGBoost	**0.78**	0.48	**0.82**	0.46
ViLD	Ensemble	0.73	0.51	0.68	0.51

Table 4.1. VilD linear probe models trained with Humanware and Awakening dataset compared with CLIP linear probe model and ViLD zero-shot

The models have shown a drop in performance when only trained on the Awakening dataset and evaluated on Humanware. This shows us that the Humanware dataset is important when training a model to detect "real-life" situations.

As we can see that these detection precision-recall scores do not meet Humanware's criteria of 0.8 precision and 0.6 recall. Instead of continuing the investigation on LP, we decided to try finetuning the image encoder of the pretrained model in RegionCLIP.

4.2. RegionCLIP

RegionCLIP experiments were conducted extensively as finetuning was available in conjunction with their pretrained models. There were two pretrained models to use for finetuning, one was trained on ResNet50 and the other on ResNet50x4. We decided to stick with ResNet50 for most of the experiments based on resource constraints as the other model

Base Model	Method	Awakening		Humanware	
		Precision	Recall	Precision	Recall
ViLD	Logistic Regression	0.77	0.48	0.67 (-0.05)	0.37 (-0.03)
ViLD	XGBoost	0.78	0.49	0.61 (-0.21)	0.42 (-0.04)
ViLD	Ensemble	0.74	0.51	0.62 (-0.06)	0.39 (-0.12)

Table 4.2. VilD linear probe models trained without Humanware dataset marked by the change in accuracy (more than 1%)

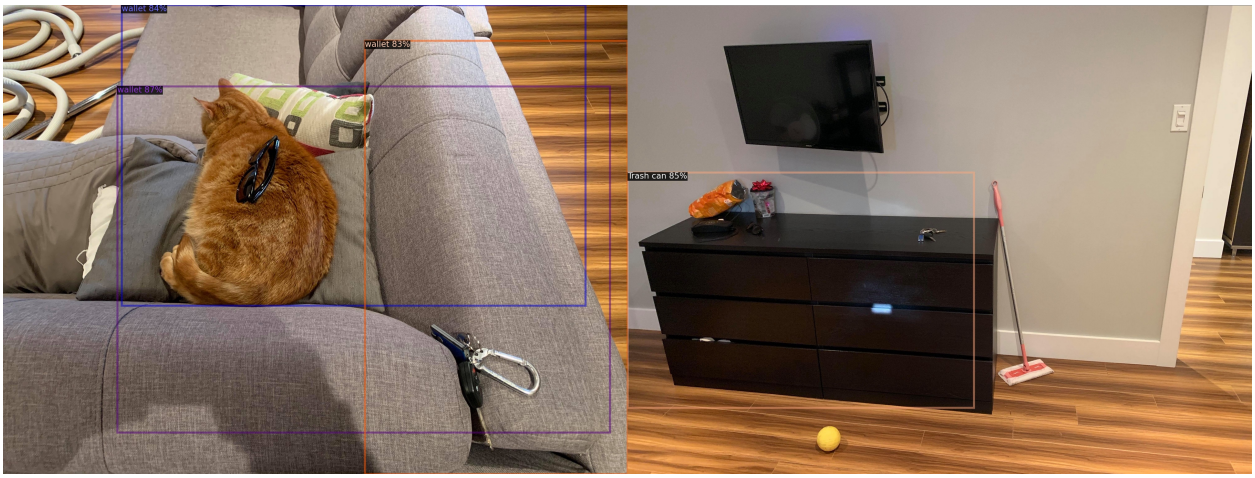


Fig. 4.2. 2 sample images from the Humanware Collected set where the model failed to detect the Keychain class.

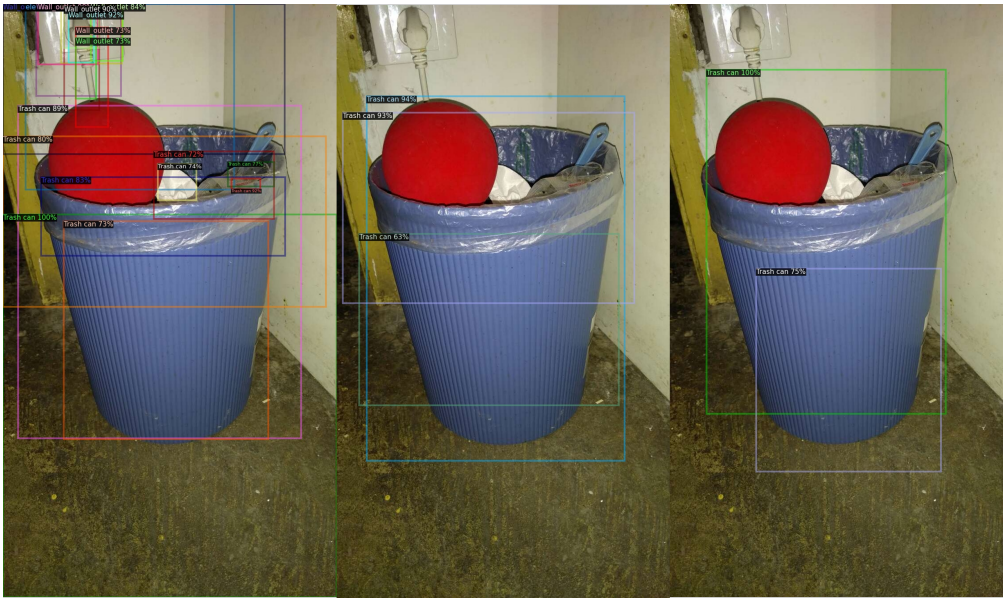
requires at least 2× more GPU memory for training on the default batch size. There were also 2 different models finetuned on COCO and LVIS datasets. We decided to first evaluate the AP on all these models on the Awakening and Humanware datasets shown in figure 1.8. The zero-shot model did not have the best mAP on either datasets and obtained 7% on the Humanware Collected dataset. The finetuned COCO and LVIS models had improved mAPs suggesting that these models, through transfer learning, were able to detect perhaps similar object concepts seen in their training set. COCO dataset finetuned model achieved 29.6% mAP and LVIS dataset finetuned model achieved 37.2% on the Humanware Collected dataset.

4.2.1. Experiments with Awakening dataset

Our first finetuned model with default parameters trained using the Awakening dataset was able to achieve 60.8% mAP on the Awakening Vector dataset and 27.1% mAP on the Humanware Collected dataset shown in table 4.4. Three examples of inference with the pretrained model on Humanware Collected data in shown in figure 4.3. Cases where the finetuned model failed to detect the Keychain class shown in figure 4.2. Being aware of these cases allows us to further investigate finetuning these classes later on.

(a) Image containing a Trash can and Keychain.

(b) Image containing a Keychain.



(c) Image containing a Trash can.

Fig. 4.3. Three Images comparing Pretrained vs COCO Transfer-Learning vs Awakening Finetuned models. The pretrained model is outputting many bounding boxes that aren't part of the relevant class resulting in low AP. The finetuned model seems to be able to detect the relevant class with fewer bounding boxes.

Table 4.3. Inference comparison of RegionCLIP's pretrained and finetuned models on Awakening and Humanware datasets. The models finetuned on COCO and LVIS performed well in terms of transfer-learning, especially on the Humanware Collected dataset. This could indicate that there are similarities in those datasets that can be generalized to the Humanware Collected dataset. The last 3 row shows the RegionCLIP pretrained/finetuned models, finetuned again on Humanware Awakening dataset. As we can see that finetuning on the Awakening dataset, reduces performance on the Humanware Collected dataset (OOD) because of overfitting.

Model	Awakening		Humanware	
	mAP	AP50	mAP	AP50
RegionCLIP Pretrained	0.034	0.127	0.074	0.166
COCO Transfer-Learning	0.281	0.441	0.296	0.413
LVIS Transfer-Learning	0.243	0.346	0.372	0.549
RegionCLIP Finetuning	0.608	0.854	0.271	0.469
COCO Finetuning	0.528	0.828	0.267	0.459
LVIS Finetuning	0.508	0.825	0.240	0.398

Table 4.4. First model finetuned on the Awakening dataset, and evaluated on Awakening and Humanware test sets with mAP of each class.

Dataset	mAP	AP50	Elevator doors	Keychain	Wallet	Trash can	Wall oulet
Awakening	0.608	0.854	0.672	0.444	0.633	0.817	0.669
Humanware	0.271	0.469	0.340	0.091	0.401	0.261	0.166

As there were many hyperparameters given by the RegionCLIP, we first decided on experimenting with settings such as text embeddings (Text prompt engineering, Feature normalization), different pretrained RPN weights (with the RPN network frozen during finetuning). For text embeddings, we tried different text prompt as there were two given, one default prompt from CLIP and another hand crafted by RegionCLIP which contains some prompts that are extended from the default as well as novel ones.

Some examples of the prompts are:

- `'a photo of the dirty {}.'`
- `'a jpeg corrupted photo of a {}.'`
- `'a {} in a video game.'`
- `'the origami {}.'`
- `'a low resolution photo of a {}.'`

After training the RegionCLIP pretrained model on the Awakening dataset, evaluating its validation set gave an mAP of 65.1% for RegionCLIP prompt embeddings and an mAP of 64.8% for CLIP prompt embeddings. These text embeddings were normalized so we also tested its non-normalized embeddings, same as the original RegionCLIP paper. The result obtained had a significant increase in mAP at 71.4% for RegionCLIP prompt embeddings

Table 4.5. The results of different finetuning RegionCLIP model with 4 different RPN Pretrained Weights on Awakening dataset.

RPN Model Weights	Awakening	
	mAP	AP50
COCO 48	**0.696**	0.875
COCO 80	0.686	**0.879**
LVIS 1203	0.334	0.466
LVIS 866 LSJ	0.680	0.875

and an mAP of 69.5% for CLIP prompt embeddings, therefore we have opted for the non-normalized RegionCLIP prompt embeddings for the rest of experiments. We have also experimented with the "Keychain" class and swapped out "Keychain" text embedding for "Key" text embedding, although these experiment results did not have a significant effect on the model performance (less than 1% in AP).

Similarly for pretrained RPN weights, we tried 4 different settings and chose the one with the highest mAP (COCO 48) evaluated on the Awakening validation set for the rest of the experiments shown in table 4.5.

4.2.2. Experiments with BasicAI and Awakening

With the arrival of BasicAI dataset, we were able to merge it with the Awakening Vector dataset and finetune the model on both datasets (Combined). First, we cross-evaluated these datasets (validation set) by training three separate models, one on Awakening, one on BasicAI, and one on both combined. Note that these are the only experiments conducted with ResNet50x4 presented with higher mAP than previous ResNet50 experiments, although they took around 40% more time to train.

The results for the model trained on the Combined datasets are shown in table 4.6. The results for the model trained on the Awakening dataset are shown in table 4.7. The results for the model trained on the BasicAI dataset are shown in table 4.8. From these comparisons, we can see that model trained on the BasicAI dataset had the highest mAP at 38% on the Humanware Collected dataset, although with a drop in performance in the Awakening dataset suggesting that it may not have properly learned the representations for certain classes without their close up images like the model trained on the Awakening dataset. The model trained on the combined datasets also was able to perform decently with an mAP of 37% on the Humanware Collected dataset. The Out of all 3 models, it was difficult to detect "Keychain" and "Wall outlets" in the BasicAI dataset suggesting that when these objects are present but small/far away (rather than large/close up in Awakening), the detection model fails to either classify it or detect it from the RPN. As we will see in the section RPN, it was found that the RPN failed to capture these classes with the default inference settings

Table 4.6. Model trained on Awakening and BasicAI (**Full**) datasets and cross-evaluated on each datasets.

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
Combined	0.696	0.744	0.506	0.875	0.688	0.670
Awakening	0.732	0.748	0.575	0.901	0.713	0.723
BasicAI	0.461	0.662	0.01	0.704	0.525	0.402
Humanware	0.372	0.500	0.110	0.464	0.412	0.373

Table 4.7. Model trained on Awakening and cross-evaluated on each dataset.

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
Combined	0.642	0.506	0.352	0.722	0.531	0.580
Awakening	0.737	0.739	0.592	0.900	0.718	0.738
BasicAI	0.260	0.696	0.002	0.466	0.02	0.115
Humanware	0.311	0.452	0.156	0.471	0.194	0.373

Table 4.8. Model trained on BasicAI and cross-evaluated on each dataset.

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
Combined	0.521	0.722	0.021	0.727	0.461	0.335
Awakening	0.413	0.697	0.026	0.747	0.353	0.243
BasicAI	0.566	0.783	0.014	0.813	0.712	0.511
Humanware	0.381	0.616	0.013	0.429	0.473	0.374

Table 4.9. The mAP values of RegionCLIP vs EfficientDet models trained on Combined and BasicAI training set evaluated on Humanware and BasicAI validation set.

Model	Humanware	BasicAI
	mAP	mAP
RegionCLIP (Combined)	0.37	0.46
EfficientDet (Combined)	0.31	0.40
RegionCLIP (BasicAI)	0.38	0.56
EfficientDet (BasicAI)	0.27	0.38

as smaller bounding boxes were filtered out. We have also requested BasicAI to relabel the "Keychain" after noticing that each individual keys on a keychain are labeled as "Keychain", this affected the detection of keychain as a whole and its bounding box size.

We have also compared the RegionCLIP models trained on the Combined and BasicAI dataset with EfficientDet-D2 [**39**] models. The results evaluated on Humanware Collected and BasicAI validation sets are shown in table 4.9. As we can see that RegionCLIP models perform better in datasets with more "real-life" scenes validating their robustness to OOD images. Some image comparisons are shown in appendix B.

PR curves for the Combined trained model evaluated on Humanware Collected and BasicAI datasets are shown in figure 4.4. We can see the difficult classes to detect are "Keychain"

Table 4.10. mAP of RegionCLIP model trained on Combined dataset evaluated on its validation set with different Focal loss parameters. First row is the default.

Parameters		mAP
γ	α	
0.25	1	0.699
0.5	1	0.695
1	1	0.696
2	1	0.695
0.5	0.5	0.694
0.5	0.25	0.685
0.25	0.25	0.690
0.25	0.1	0.675

and "Wall outlet" on both datasets, although with a higher confidence score set, "Wall outlet" can reach a higher precision (more than 80%) at a cost of lower recall (less than 50%) which is not the case for "Keychain". Trash cans and wallets also seem to have a lower precision, especially at higher recall values suggesting many false positives. A more detailed examination of these curves is shown in appendix D.

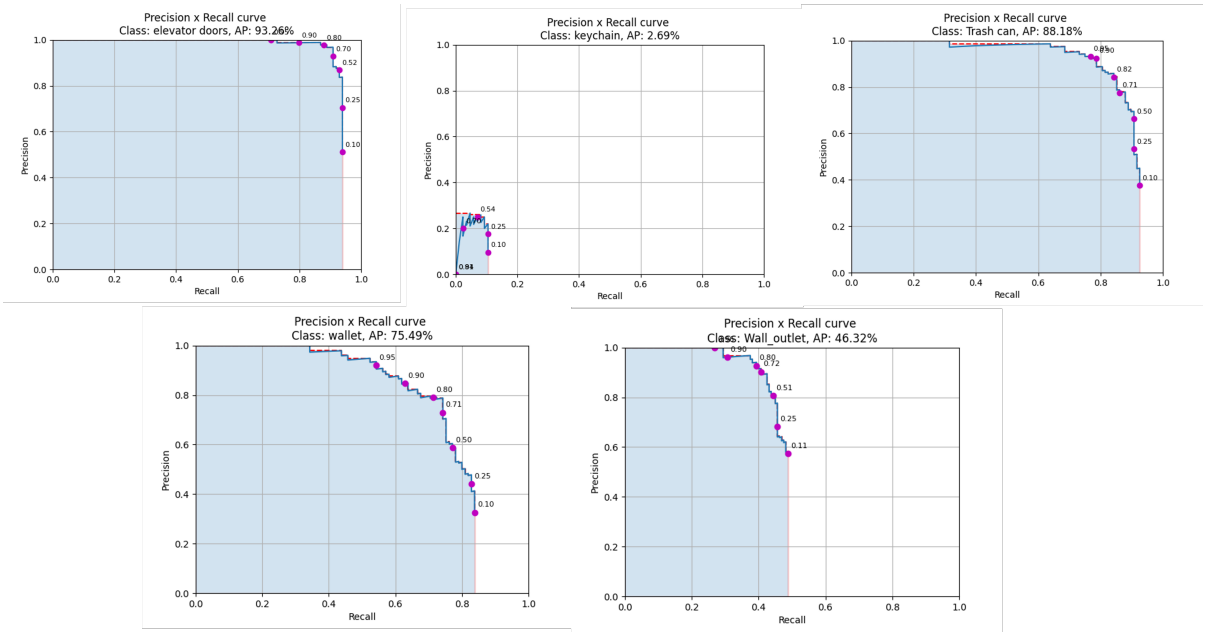
4.2.3. Focal loss

A small grid search was conducted on focal loss parameters when training the model with the Combined dataset. Their mAP scores on the Combined validation set are shown in table 4.10.

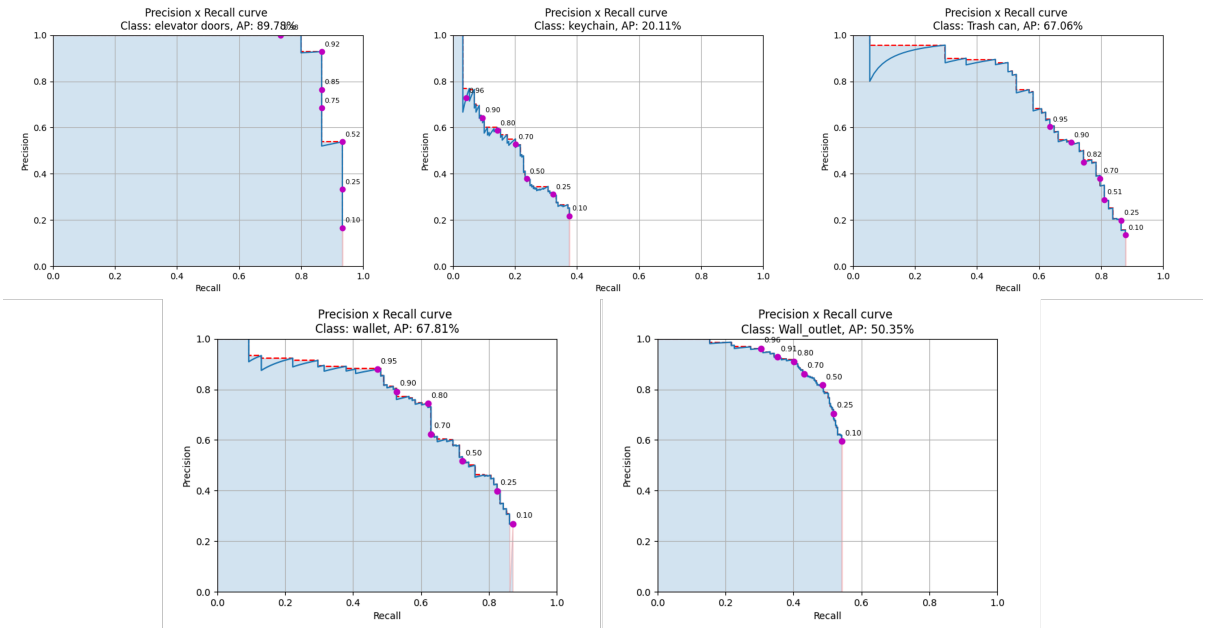
There doesn't seem to be a significant change in mAP between the different γ (Gamma) values though smaller gamma values (less than 0.5) are slightly more preferred shown by the trend. Similarly, for α (Alpha) values, it is slightly preferred to have higher values. There may be many difficult (OOD) samples for the model to learn and not enough samples to learn from which is why changing the focal loss parameters didn't have much of an effect. The default parameters, 0.25 γ & 1 α are kept in the end.

4.2.4. RPN

RegionCLIP inference configuration included RPN settings such as NMS and objectness score pre and post NMS bounding box filter. We investigated a combination of these set of configurations graphed by the percent of bounding boxes missed by the RPN compared with the GT for a model trained on the Combined (BasicAI and Awakening) dataset for each individual class shown in figure 4.5. As you can see that many of the "Keychain" and "Wall outlet" samples were missed by the RPN. The best setting that is able to capture most of them is 0.7 NMS, 24k pre and 4k post objectness score bounding box filter. Although the inference time for this setting is at least double the next best setting with 0.7 NMS, 12k pre



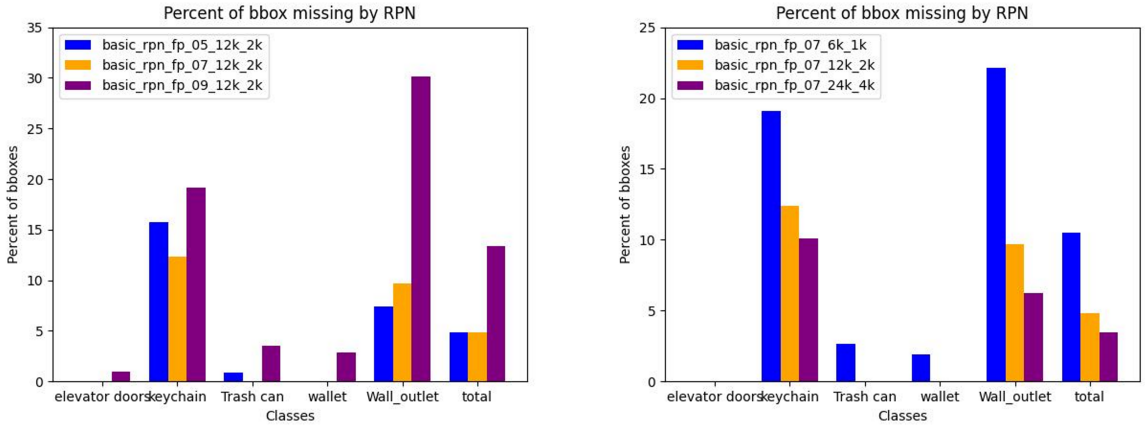
(a) PR Curve on BasicAI test set.



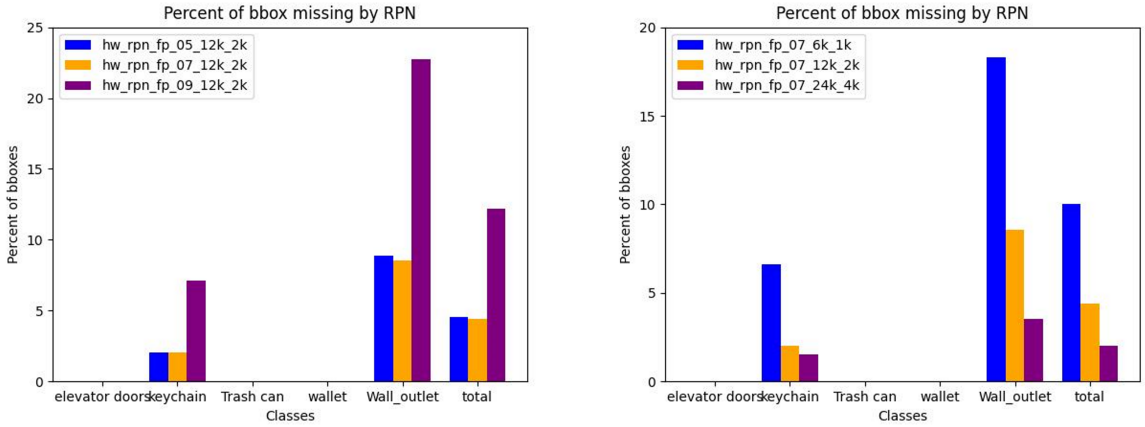
(b) PR Curve on Humanware Collected test set.

Fig. 4.4. Precision-Recall Curves of the model trained on Combined dataset generated for each class with their respective mAPs.

and 2k post objectness score filter, therefore we opted for the ladder instead for the rest of the experiments. Some images of the RPN bounding box outputs after updating the settings with statistics on the overlap with the GT bounding boxes are shown in appendix C.



(a) % Missed bounding boxes on BasicAI test set



(b) % Missed bounding boxes on Humanware Collected test set

Fig. 4.5. Percentage of GT bounding boxes missed by the RPN graphed for each individual class on 6 different settings. Each setting are defined by the last three numbers as "[NMS]_[PRE]_[POST]" filters shown in the legend.

Table 4.11 shows the difference in terms of mAP performances after updating the RPN inference settings with the model trained on the Combined dataset with updated hyperparameters and newly labelled "Keychain" in the BasicAI dataset discussed previously.

We also tested finetuning the model with RPN weights unfrozen (with and without the RPN vision encoder backbone weights unfrozen). The results are shown in table 4.12. It seems that finetuning the RPN didn't help much in terms of overall performance. We hypothesized that the RPN tried to average the gap in terms of its bounding box predictions by compensating for the classes that have lower bounding box sizes during detection at the cost of disregarding larger bounding box classes such as "elevator doors".

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
BasicAI	0.58 (+0.06)	0.78 (+0.02)	0.30 (+0.02)	0.71	0.53	**0.40 (+0.11)**
Humanware	0.42 (+0.03)	0.61 (+0.02)	0.22 (+0.03)	0.46 (-0.01)	0.41 (+0.01)	**0.40 (+0.09)**

Table 4.11. The AP values shows the difference compared to the previous best model trained on the Combined dataset. Significant improvements on the "Wall outlet" class are in bold.

Backbone	Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall outlet
Frozen	BasicAI	0.52	0.73 (-0.03)	0.29 (+0.01)	0.71	0.53	0.31 (+0.02)
Frozen	Humanware	0.39	0.56 (-0.03)	0.20 (+0.01)	0.48 (+0.01)	0.40	0.32 (+0.01)
Unfrozen	BasicAI	0.52	0.74 (-0.02)	0.29 (+0.01)	0.71	0.54 (+0.01)	0.32 (+0.03)
Unfrozen	Humanware	0.39	0.59	0.19	0.47	0.40	0.32 (+0.01)

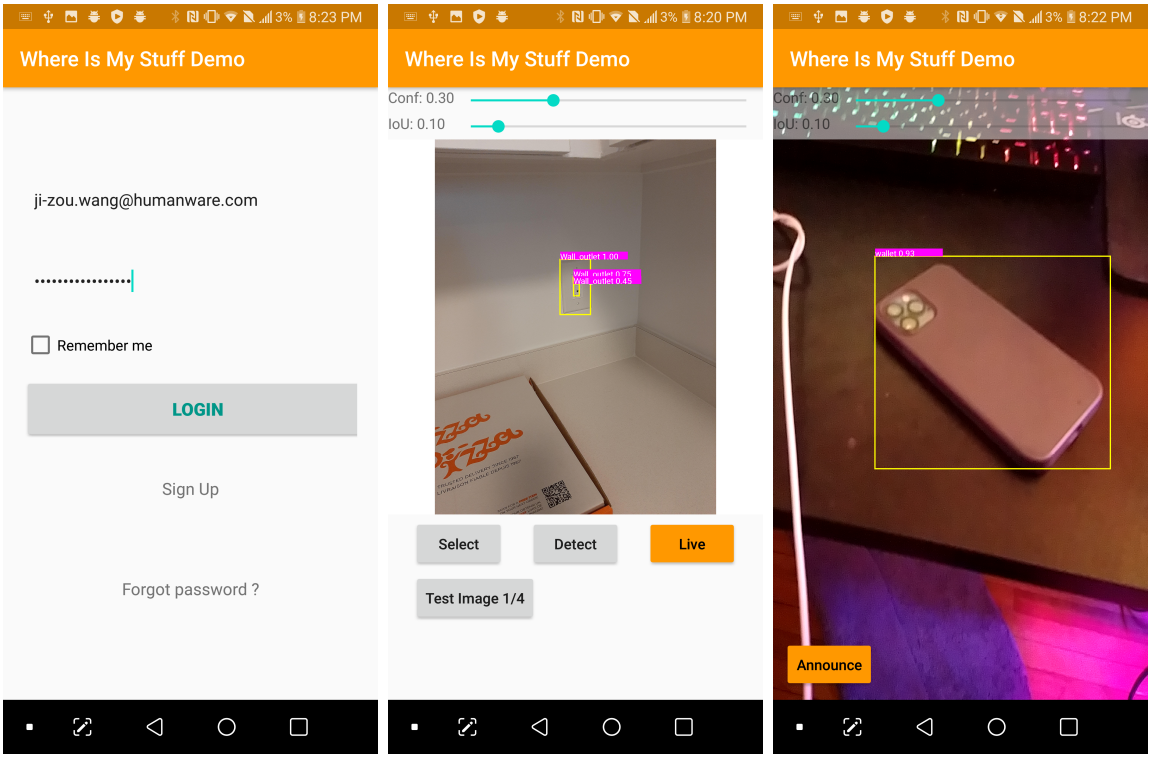
Table 4.12. A comparison of RPN finetuned model, with frozen and unfrozen RPN backbone. The AP values shows the difference compared to the previous best model trained on the Combined dataset.

4.2.5. Android Application

From the models trained in the experiments, we have developed a proof of concept android application using AWS (Amazon web service) EC2 (Elastic Compute Cloud) service which provides the cloud compute capacity to host our demo server. The AWS server hosted a Tesla M60 GPU. It was a step down compared to the Humanware server's GeForce RTX 3090 GPU, at least 3× lower in terms of compute power based on Passmark scores which resulted in slower inference times.

The demo server was developed using Flask [**14**] a lightweight web framework written in Python. The server pre-loads the best performing RegionCLIP model (on the Humanware Collected test set) with a default confidence score threshold and IoU for NMS. The client can then send a request with an image (in static mode) or a series of images (in live mode) to the model which then computes the bounding boxes' location and class of the detected objects.

The demo client application was built on android and tested on an LG G6 mobile device. The user interface and inference images are shown in figure 4.6. The current client allows the change of confidence score threshold (Conf) and IoU values applied to the detected bounding boxes from the server. You can also select images previously taken from the mobile device for inference or just cycle through a list of 4 test sample images. When using live mode by



(a) User Login page. Users can sign up with their email with account details stored on the AWS server.

(b) Static mode detection on a test image detecting "wall outlets".

(c) Live mode detection showing a false positive, a "mobile phone" detected as a "wallet".

Fig. 4.6. "Where is my stuff" Android Application Demo.

pressing on the "Live" button, one can also press the "Announce" button to verbally locate the detected object shown in the image. The location is referenced by the size and the location of the bounding box shown in the image. For example, a small bounding box on the lower right would announce: "an [object class] is detected far from you on your lower right". Knowing that RegionCLIP is a two-stage detector, each inference takes on average 5 seconds to process per image from the AWS server. Although this speed can be improved with a faster GPU as tested on the Humanware server which took on average 2 seconds to process per image.

We used the application to test out the models in real-life situations helping us evaluate points of success and failure with many potential false positive cases for qualitative analysis.

4.2.6. Experiments with False positive class labels

Based on the qualitative analysis of test set images from BasicAI and Humanware Collected datasets using T-SNE Visualizations with some examples shown in appendix A as well

as inference using the Android Application, we were able to compile a list of false positive classes.

Before any training or labelling on these false positive classes on our datasets, we wanted to see if just adding a few of these class labels on top of the original 5 classes would affect the inference of our model in a zero-shot manner. When tested with false positive classes on "Trash cans" such as "cup", "pot", "can", "bottle", and "bowl", the results varied and were not conclusive. Adding certain classes such as "cup", "can", and "bottle" can help improve the performance of detecting "Trash can" while having the other classes such as "pot" and "bowl" impair the performance of detecting "Trash cans". We concluded that the model may not have learned a clear separation between their class labels and the detected object therefore needs finetuning on these classes to better separate them.

A list of these false positive classes was generated for finetuning on the BasicAI dataset which was mentioned in section 2.3. We also used COCO Indoor dataset to generation additional false positive examples and classes given our limited examples from our own datasets. These models were trained on a combination of BasicAI and COCO Indoor dataset and evaluated on the BasicAI dataset shown in table 4.13 and Humanware Collected dataset shown in table 4.14. A few image comparisons between the model trained on the BasicAI 5 (original 5 classes) vs BasicAI 63 & Coco Indoor (additional false positive classes) are shown in figure 4.7.

Some of the models were trained with LSJ data augmentation which resulted in better performance on all class labels other than "elevator doors". We hypothesized that it would be better to filter out "elevator doors" before LSJ augmentations as the model may have learned a different visual representation for the ones that are augmented by LSJ, therefore evaluating "real-life" scenarios where the test images are "elevator doors" close in proximity would still have a large bounding box. Using this idea, indeed the performance with the LSJ (filter) resulted in better AP on "elevator doors". In the general, having to train multiple false positive class labels can sometimes help improve the AP performance of the relevant classes, especially with BasicAI 63 on BasicAI test set but not much on the Humanware Collected test set. It is not conclusive if finetuning on false positive classes helps in performance as the difference in mAP fluctuates based on the number of false positive class samples in the dataset. More false positive samples may be required for finetuning to better separate samples that are closer in the embedding space.

4.2.7. Other experiments

We tried training with a simple linear probe by disabling the last text embedding layer and setting it as a simple linear classification layer in the network on the Awakening and BasicAI datasets. The mAP performance had a reduction of 30% on the validation set. We

(a) Image showing "Trash can" false positives on the left and correctly classified as "bowl" on the right

(b) Image showing "Trash can" false positives on the left and correctly classified with the additional false positive classes such as "cup", "kitchen_pot", "food_jar", on the right.

Fig. 4.7. Image comparison between BasicAI 5 (Left) vs (Right) Basic 63 & Coco Indoor with extra false positive class labels.

also tried training with the last text embedding layer unfrozen, this didn't improve the mAP performance either. As the RegionCLIP authors intended, the text embeddings layer should be kept frozen with the model training focused on aligning the region embeddings to the text embeddings, especially for open set classification.

4.3. Problems Encountered & Challenges

While the experiments presented above showed interesting results, it is difficult to tell the statistical significance without training multiple models and obtaining their confidence interval of the AP values to hypothesis test if a different parameter had a significant change

Table 4.13. Evaluation results on BasicAI test set with models trained on the dataset specified in the dataset column with or without LSJ data augmentation. Note, BasicAI 5 is the BasicAI with the original 5 class labels, and BasicAI 63 is with an additional 63 false positive labels. The best results are bolded.

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
BasicAI 5	0.594	0.744	0.358	0.758	0.612	0.497
BasicAI 5 LSJ	0.582	0.563	0.390	0.771	0.637	0.547
BasicAI 5 & Coco	0.578	0.713	0.302	0.750	0.614	0.511
BasicAI 63	0.672	**0.837**	0.407	0.828	**0.749**	0.540
BasicAI 63 LSJ	0.646	0.641	0.450	0.838	0.739	0.564
BasicAI 63 LSJ (filter)	**0.687**	0.827	**0.456**	**0.834**	0.746	**0.571**
BasicAI 63 & Coco	0.636	0.774	0.381	0.806	0.708	0.512

Table 4.14. Evaluation results on Humanware Collected test set with models trained on the dataset specified in the dataset column with or without LSJ data augmentation. The best results are bolded.

Dataset	mAP	Elevator doors	Keychain	Trash can	Wallet	Wall oulet
BasicAI 5	0.389	0.588	0.216	0.370	0.401	0.367
BasicAI 5 LSJ	**0.416**	0.502	**0.305**	0.404	**0.434**	**0.433**
BasicAI 5 & Coco	0.402	0.622	0.242	0.387	0.384	0.374
BasicAI 63	0.389	0.570	0.182	**0.424**	0.398	0.370
BasicAI 63 LSJ	0.394	0.486	0.218	0.419	0.421	0.426
BasicAI 63 LSJ (filter)	0.401	0.512	0.228	0.412	0.429	0.424
BasicAI 63 & Coco	0.401	**0.629**	0.197	0.375	0.425	0.381

compared to the referenced parameter. This requires a lot more time and resource as training each model took on average 2-3 days.

During training for each model on RegionCLIP, we have also encountered "NaN loss" errors which was resolved by adapting the learning rates at different epochs before the "NaN loss" error occurred. Many adaptations were also made to the RegionCLIP code for training and inference on Humanware datasets that allowed us to customize hyperparameters that wasn't originally part of RegionCLIP for experimentation such as different optimizers.

Chapter 5

Conclusion and Future Work

From the results of the experiments on VilD and RegionCLIP, we can see that these foundation models have good capabilities when it comes to open-class detection. After finding the optimal hyperparameters and RPN configurations for RegionCLIP, it was able to classify OOD objects with greater mAP performance than a non-foundation model counterpart, EfficientDet. Perhaps when pretrained on even more data shown with neural scaling laws [**21**], these models can outperform a finetuned model in open class detection. Although our task at hand mainly focuses on class concepts that are well defined, therefore the model is able to better adapt after finetuning achieving high AP performance even on an OOD dataset. The OOD detection performance also depends on the dataset that Humanware has. As more and more data are collected and finetuned, a "strong" benchmark dataset would be required to measure OOD performance which could be obtained through active learning.

In terms of incorporating additional false positive classes in the dataset as described in section 4.2.6, there is a heavy sample imbalance which can introduce a bias during training. This can affect the inference results and learning a proper separation between the false positive classes and the original 5 classes. A suggestion can be to only include false positive classes meeting a certain number of sample thresholds or to obtain more class samples and attain an mAP threshold for detecting the false positive class when evaluated on the validation set. It may be costly to incorporate all the data samples of a false positive class as more false positive classes emerge.

5.1. RegionCLIP

Towards the end of the internship, RegionCLIP released pretrained object detection models based on UniCL [**42**] a unified image-label-text representation contrastive learning objective that was used in Florence [**43**]. It would be interesting to perform zero-shot and finetuning experiments with this new pretrained model as it was able to achieve 14.5% over

supervised learning methods in zero-shot classification accuracy and 3.4% in linear probe settings.

Used in conjunction with UniCL, K-LITE [**37**], a simple proposed pretraining strategy that uses external knowledge through enriching text with WordNet and Wiktionary to learn image-text representations. The external knowledge is appended to the original text concepts. It was able to outperform the top pretrained UniCL classification model in zero-shot task benchmarks. Generating text concepts for Humanware's classes using this proposed method could potentially help alleviate false positive detections as it may clarify the classification boundary based on additional textual descriptions of the object class.

5.2. Yolov7

Yolov7 [**40**] a recently released incremental update to the Yolo [**35**] model family is able to outperform, at the time of release, all known real-time (30 FPS or higher) object detectors in speed and accuracy achieving state-of-the-art results. It incorporates module re-parameterization of different networks by analyzing gradient flow propagation paths and aggregation in term of E-ELAN [**40**], an extended efficient layer aggregation network which applies group convolution to expand the channel and set of computational blocks with the same parameters. These features map outputs are shuffled and merged to enhance the features learned by multiple mappings. Similiar to EfficientNet [**38**], it uses a compound model scaling method while maintains the original architecture structure.

This model can be used as a new baseline compared with the pretrained foundation models for open-class object detection. It also has a much faster inference speed therefore it would be more practical to deploy this model if there isn't a huge gap in terms of AP performance on the OOD benchmark dataset compared to foundation models.

5.3. Head2Toe

An interesting alternative to linear probe called Head2Toe [**11**] is able to match finetuning performance, especially for OOD transfer learning as well as reducing training and storage cost by a factor of 100. This algorithm allows linear probe access to features from all layers, rather than only the top-layer features. These layers are aggregated using average pooling [**2**] to reduce dimensionality and access to these features is sparsely regularized based on group lasso [**44**] to select a few of the most useful features within the network for learning. This algorithm can be adapted to the proposed foundation models such as RegionCLIP further enhancing OOD robustness compared to finetuning.

5.4. Other suggestions and Ethics Discussion

3D object detection using LiDAR point cloud sensors has been presented in many applications such as self-driving cars. Research on detecting 3D point cloud objects using attention [**29**] and transformer model [**30**] benchmarked on ScanNet [**6**] shows incredible progress. This suggests further investigation on the implementation of a 3D object detector for our task if LiDAR sensors are available.

Other foundation models such as Florence [**43**] and Flamingo [**1**] can also be considered for future research as they hold great potential to generalize to many tasks within the vision-language domain. This can allow object localization without an explicit object detector network through visual reasoning, visual language navigation [**31**] or VQA. A caveat to note is that the benchmark dataset must be broad and "heavily" tested as the model itself may exhibit unpredictable behaviour.

Foundation models are at the heart of current research but also have their own biases as they are trained with large public datasets. One must be aware of the ethical concerns before implementing these models in their applications. One must strive for AI alignment when it comes to general foundation models to prevent harm to its users, guiding the user with safety in mind towards the object of interest.

5.5. Learnings

Overall, I have gained a broad understanding of object detection methods through experiments with classical and foundation object detection models. The experiments have allowed me to understand the pipeline that goes into training and deploying an object detector from data labelling, preprocessing, and analysis to model finetuning, evaluation and monitoring using a demo application. Collaborating with the team through daily meetings, I have been able to inquire and brainstorm ideas while implementing and managing resources shared with others.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman et Karen Simonyan : Flamingo: a visual language model for few-shot learning, 2022.

[2] Florentin Bieder, Robin Sandkühler et Philippe C. Cattin : Comparison of methods generalizing max- and average-pooling, 2021.

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa et Larry S. Davis : Soft-nms – improving object detection with one line of code, 2017.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever et Dario Amodei : Language models are few-shot learners, 2020.

[5] Tianqi Chen et Carlos Guestrin : XGBoost. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser et Matthias Niessner : Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017.

[7] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan et Lei Zhang : Dynamic head: Unifying object detection heads with attentions, 2021.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit et Neil Houlsby : An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[9] Zi-Yi Dou, Yichong Xu, Zhe Gan, Jianfeng Wang, Shuohang Wang, Lijuan Wang, Chenguang Zhu, Pengchuan Zhang, Lu Yuan, Nanyun Peng, Zicheng Liu et Michael Zeng : An empirical study of training end-to-end vision-and-language transformers, 2021.

[10] Linus Ericsson, Henry Gouk, Chen Change Loy et Timothy M. Hospedales : Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, may 2022.

[11] Utku Evci, Vincent Dumoulin, Hugo Larochelle et Michael C. Mozer : Head2toe: Utilizing intermediate representations for better transfer learning. 2022.

[12] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le et Barret Zoph : Simple copy-paste is a strong data augmentation method for instance segmentation, 2020.

[13] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang et Quoc V. Le : Nas-fpn: Learning scalable feature pyramid architecture for object detection, 2019.

[14] Miguel Grinberg : *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[15] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo et Yin Cui : Open-vocabulary object detection via vision and language knowledge distillation. 2021.

[16] Agrim Gupta, Piotr Dollár et Ross Girshick : Lvis: A dataset for large vocabulary instance segmentation, 2019.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár et Ross Girshick : Mask r-cnn, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun : Deep residual learning for image recognition, 2015.

[19] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li et Tom Duerig : Scaling up visual and vision-language representation learning with noisy text supervision. 2021.

[20] Glenn Jocher : YOLOv5 by Ultralytics, 5 2020.

[21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu et Dario Amodei : Scaling laws for neural language models, 2020.

[22] Diederik P. Kingma et Jimmy Ba : Adam: A method for stochastic optimization, 2014.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie : Feature pyramid networks for object detection, 2016.

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He et Piotr Dollár : Focal loss for dense object detection, 2017.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick et Piotr Dollár : Microsoft coco: Common objects in context, 2014.

[26] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi et Jiaya Jia : Path aggregation network for instance segmentation, 2018.

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C. Berg : SSD: Single shot MultiBox detector. *In Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin et Baining Guo : Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[29] Ze Liu, Zheng Zhang, Yue Cao, Han Hu et Xin Tong : Group-free 3d object detection via transformers. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2949–2958, 2021.

[30] Ishan Misra, Rohit Girdhar et Armand Joulin : An end-to-end transformer model for 3d object detection. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

[31] Sang-Min Park et Young-Gab Kim : Visual language navigation: a survey and open challenges. *Artificial Intelligence Review*, pages 1–63, 2022.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay : Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger et Ilya Sutskever : Learning transferable visual models from natural language supervision, 2021.

[34] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen et Ilya Sutskever : Zero-shot text-to-image generation, 2021.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi : You only look once: Unified, real-time object detection, 2015.

[36] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[37] Sheng Shen, Chunyuan Li, Xiaowei Hu, Jianwei Yang, Yujia Xie, Pengchuan Zhang, Zhe Gan, Lijuan Wang, Lu Yuan, Ce Liu, Kurt Keutzer, Trevor Darrell, Anna Rohrbach et Jianfeng Gao : K-lite: Learning transferable visual models with external knowledge, 2022.

[38] Mingxing Tan et Quoc V. Le : Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[39] Mingxing Tan, Ruoming Pang et Quoc V. Le : Efficientdet: Scalable and efficient object detection. 2019.

[40] Chien-Yao Wang, Alexey Bochkovskiy et Hong-Yuan Mark Liao : Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[41] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo et Ross Girshick : Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[42] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Bin Xiao, Ce Liu, Lu Yuan et Jianfeng Gao : Unified contrastive learning in image-text-label space, 2022.

[43] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou et Pengchuan Zhang : Florence: A new foundation model for computer vision, 2021.

[44] Ming Yuan et Yi Lin : Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

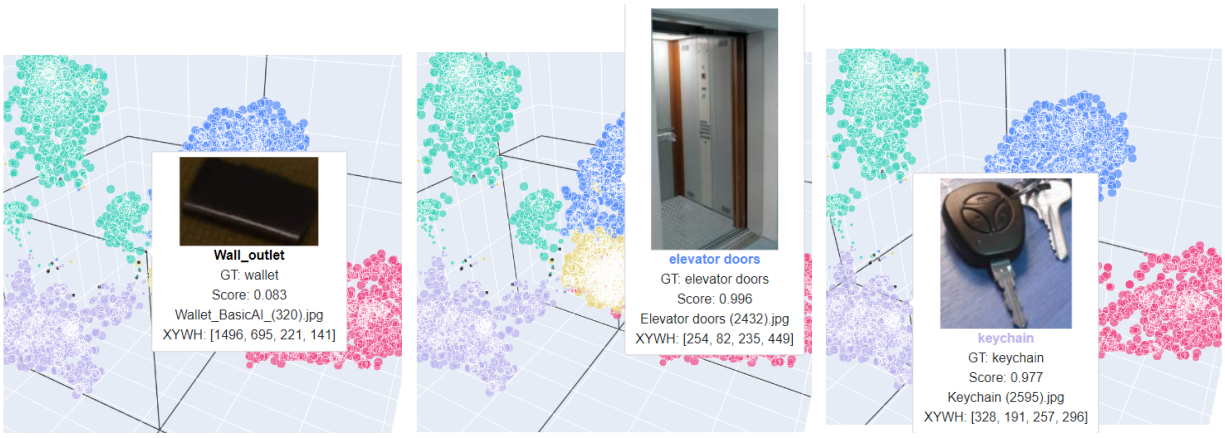
[45] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li et Jianfeng Gao : Regionclip: Region-based language-image pretraining, 2021.

Appendix A

---

t-SNE Visualizations

Additional examples of the dash application built to visualize the region feature embedding outputs using t-SNE.



(a) A "wallet" classified as a "Wall outlet".

(b) A correct "Elevator doors" classification.

(c) A correct "Keychain" classification.

Fig. A.1. GT region samples showing the classified class in colour, GT class, Confidence score, file name of the image and the bounding box location on the image.

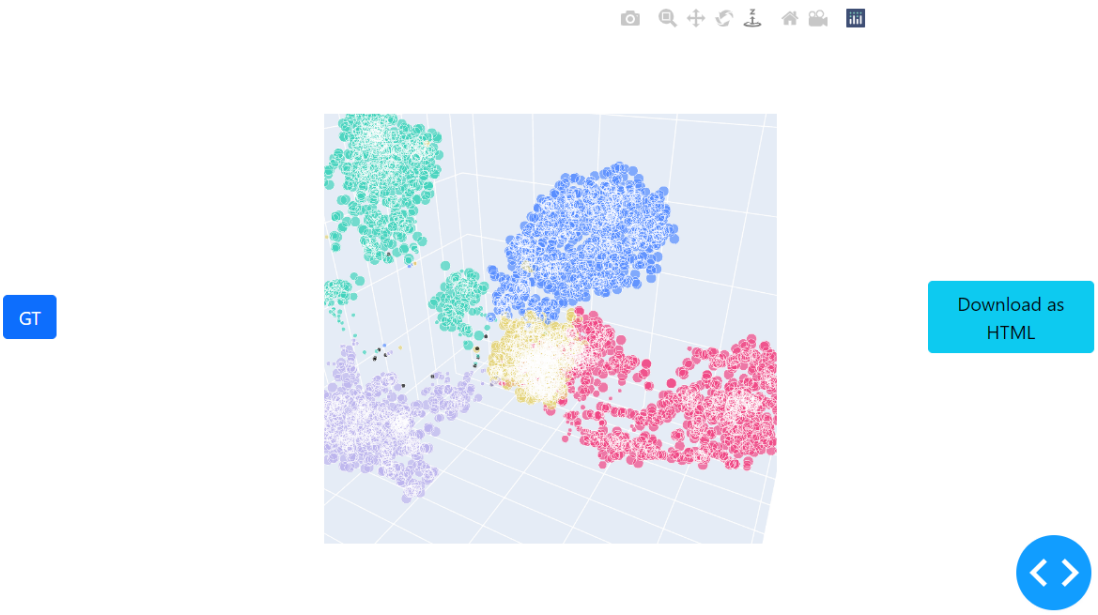
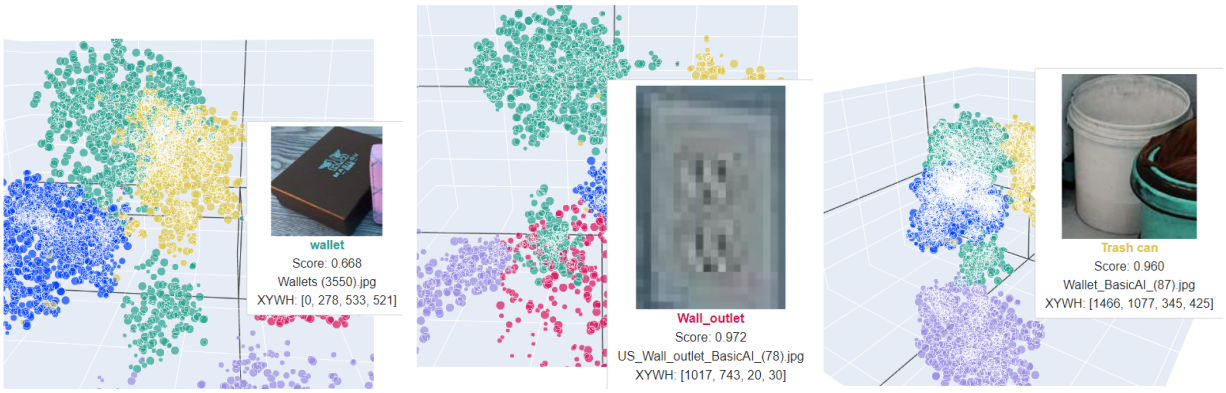


Fig. A.2. GT region feature embedding t-SNE visualization. The 5 different coloured dots show the 5 classes that are correctly labelled and the black dots show incorrectly labelled classes relative to the GT label. Button on the right to toggle between GT and RPN region feature embeddings. Button on the left to download the plot as HTML file.



(a) A false positive "wallet" classification.

(b) A correct "Wall outlet" classification. High pixelation as it's a small bounding box region.

(c) A false positive "Trash can" classification.

Fig. A.3. RPN region samples showing the classified class in colour, Confidence score, file name of the image and the bounding box location on the image.

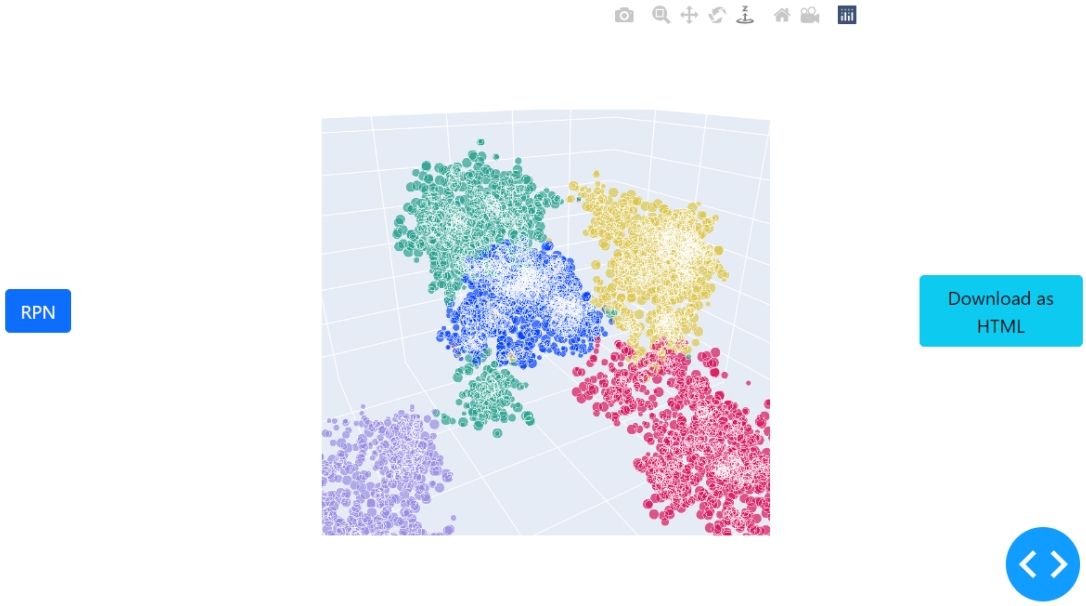


Fig. A.4. RPN region feature embedding t-SNE visualization. The 5 different coloured dots shows the 5 labeled classes. Button on the right to toggle between GT and RPN region feature embeddings. Button on the left to download the plot as html file.

Appendix B

Efficient-D2 vs RegionCLIP model inferences

Images comparing detections from EfficientDet-D2 (Left) vs RegionCLIP (Right) models trained on the Combined Awakening Vector and BasicAI datasets.

Fig. B.1. Both models are able to detect the trash can in the image.

Fig. B.2. RegionCLIP model was able to detect the trash can in the image but also a false positive.

Fig. B.3. RegionCLIP model was able to detect the wall outlet with a high confidence false positive trash can.

Fig. B.4. RegionCLIP model failed to detect the wall outlet compared to EfficientDet-D2 with many false positives but was able to detect the trash can.

Appendix C

RPN Visualizations

Visualizations showing the before and after of using different RPN inference settings, from 0.9 NMS, 6k pre and 1k post objectness score bounding box filter to 0.7 NMS, 12k pre and 2k post objectness score bounding box filter.

The bounding box proposals generated by the RPN are highlighted in purple and the GT bounding boxes are highlighted in white. Statistics on the GT bounding in white, from top to bottom box show the number of RPN bounding boxes overlapping with the GT, the number of RPN bounding boxes that overlap but also are the top predicted class out of all the other classes and lastly, those that also has a prediction confidence score of at least 50%. If there are RPN boxes that overlap with the GT, the statistics in yellow show the highest detected confidence score with its IoU and the highest detected IoU with its confidence score of that class.

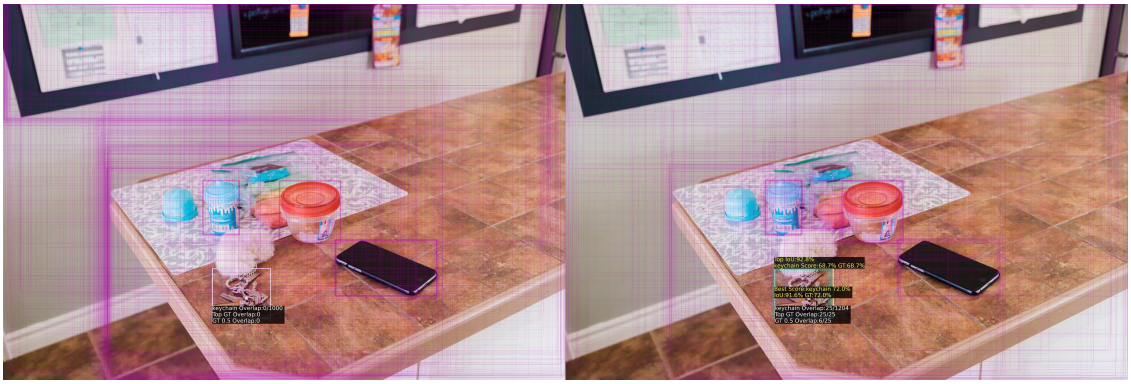


Fig. C.1. A keychain on the table previously not able to be detected by the RPN.

Fig. C.2. A keychain on the bed previously not able to be detected by the RPN.

Fig. C.3. A wall outlet previously not able to be detected by the RPN.

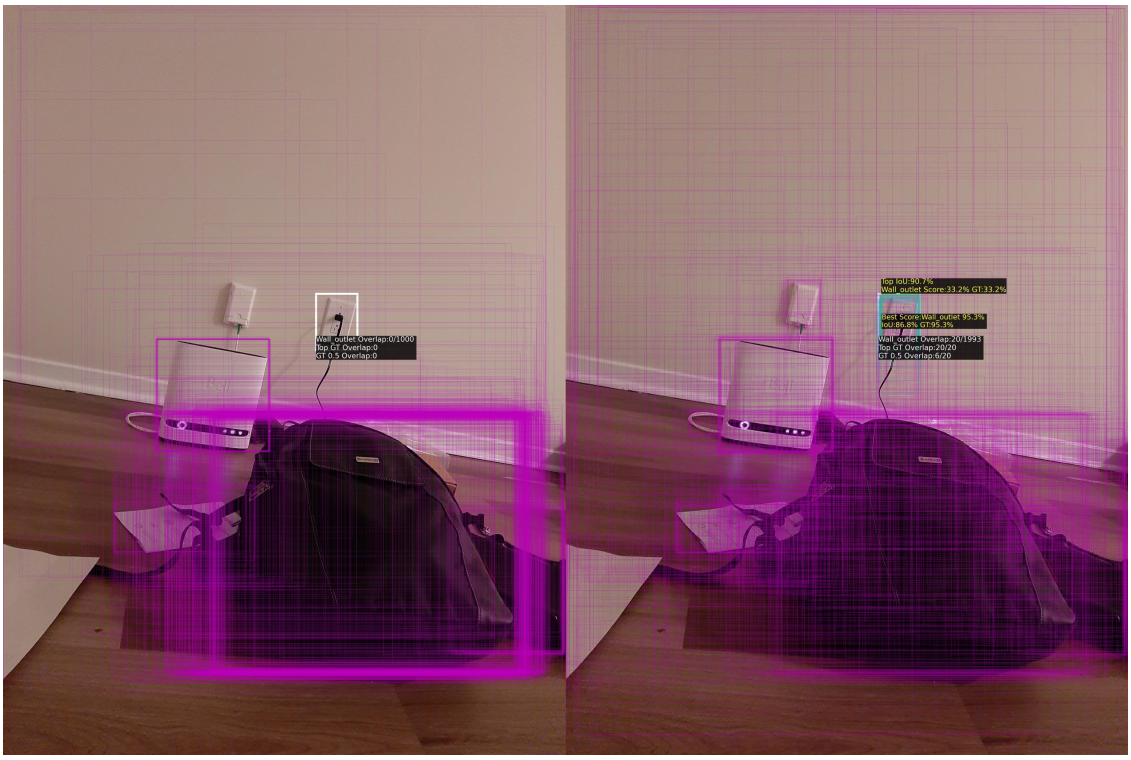


Fig. C.4. A wall outlet previously not able to be detected by the RPN.

Appendix D

Precision Recall Visualizations

We tried to visualize and measure the cutoff classification confidence score which would end up meeting Humanware's criteria of 60% recall and 80% precision on the Humanware Collected shown in D.1 and BasicAI dataset shown in D.2.

Table D.1. Given a set of Precision values (ranging from 0.7 to 1.0), give the recall and classification confidence scores (Score) of all 5 classes. The bolded confidence score is the lowest score threshold that meets Humanware's criteria of precision (0.8) and recall (0.6). As seen, it is difficult for most classes other than "Elevator doors" to achieve these criteria on the Humanware Collected dataset. This table refers to the PR curve shown in D.1.

Precision	Elevator doors		Keychain		Trash can		Wallet		Wall outlet	
	Recall	Score	Recall	Score	Recall	Score	Recall	Score	Recall	Score
1.00	0.800	0.936	0.027	0.982	0.013	1.000	0.009	0.999	0.145	0.991
0.95	0.867	0.922	0.032	0.972	-	-	-	-	0.304	0.958
0.90	-	-	-	-	-	-	0.315	0.980	0.401	0.805
0.80	0.867	**0.875**	-	-	0.500	0.991	0.519	0.914	0.485	0.503
0.70	0.867	0.799	0.037	0.962	0.622	0.971	0.630	0.702	0.517	0.250

Table D.2. The precision-recall criteria are achieved on most classes in the BasicAI dataset other than "Keychain" (no precision above 70%) and "Wall outlet". This table refers to the PR curve shown in D.2.

Precision	Elevator doors		Keychain		Trash can		Wallet		Wall outlet	
	Recall	Score	Recall	Score	Recall	Score	Recall	Score	Recall	Score
1.00	0.687	0.970	-	-	0.009	1.000	0.286	0.990	0.200	0.971
0.95	0.859	0.868	-	-	0.732	0.972	0.371	0.982	0.356	0.910
0.90	0.939	**0.510**	-	-	0.815	0.907	0.429	0.971	0.419	0.707
0.80	-	**-**	-	-	0.833	**0.803**	0.610	**0.862**	0.450	0.512
0.70	0.950	0.251	-	-	0.861	0.557	0.629	0.752	0.469	0.276

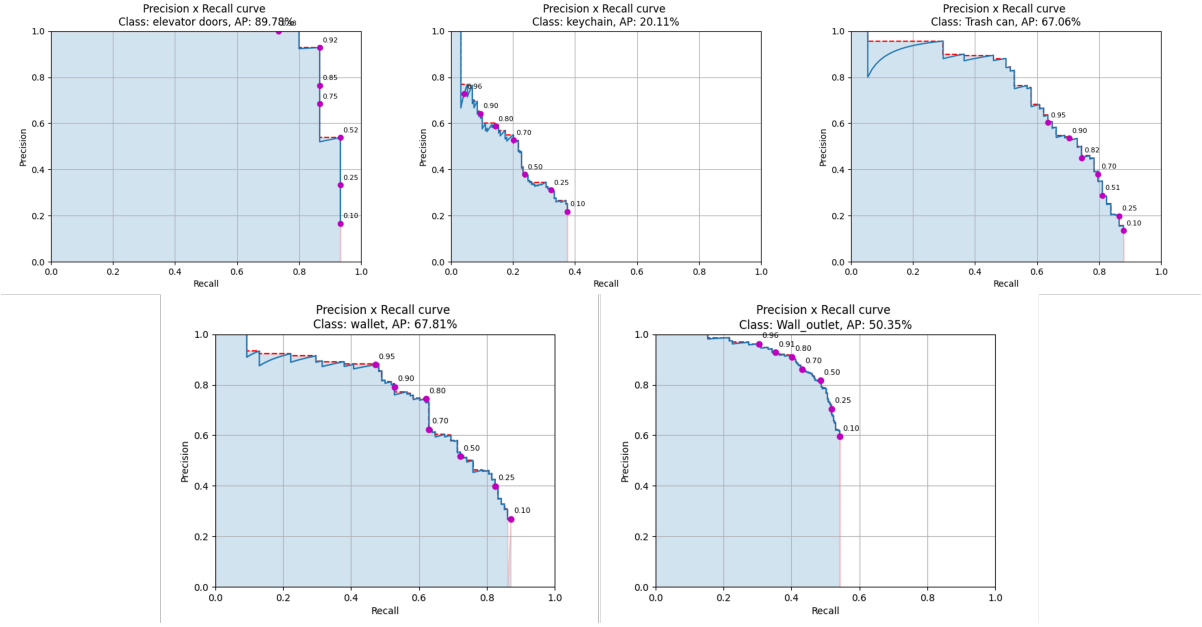


Fig. D.1. PR Curve on Humanware Collected test set.

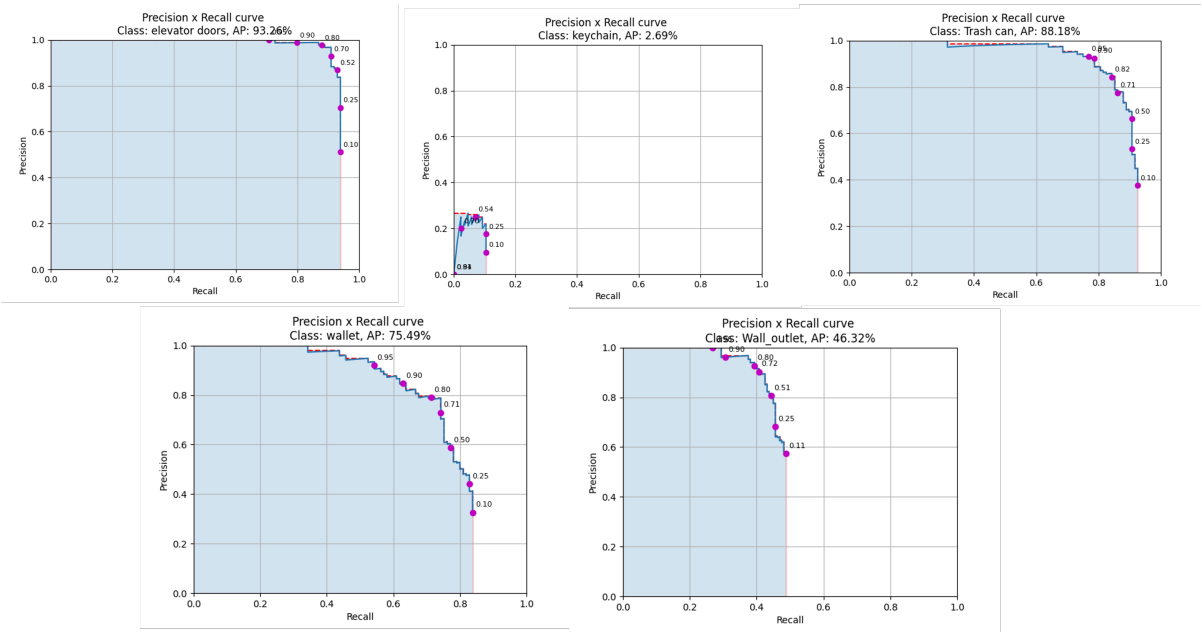


Fig. D.2. PR Curve on BasicAI test set.

Ordre des éléments constitutifs du mémoire ou de la thèse		
1.	La page de titre	obligatoire
2.	La page d'identification des membres du jury	obligatoire
3.	Le résumé en français et les mots clés français	obligatoires
4.	Le résumé en anglais et les mots clés anglais	obligatoires
5.	Le résumé dans une autre langue que l'anglais ou le français (si le document est écrit dans une autre langue que l'anglais ou le français)	obligatoire
6.	Le résumé de vulgarisation	facultatif
7.	La table des matières, la liste des tableaux, la liste des figures ou autre	obligatoires
8.	La liste des sigles et des abréviations	obligatoire
9.	La dédicace	facultative
10.	Les remerciements	facultatifs
11.	L'avant-propos	facultatif
12.	Le corps de l'ouvrage	obligatoire
13.	Les index	facultatif
14.	Les références bibliographiques	obligatoires
15.	Les annexes	facultatifs
16.	Les documents spéciaux	facultatifs