ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Enhanced Object Proposal Generation for Weakly Supervised Instance Segmentation

Master Thesis

K. Umar

September 13, 2023

Advisors: Prof. Dr. C. Zhang
Department of Computer Science, ETH Zürich

Abstract

Machine Learning has benefited immensely from scaling datasets both in terms of size and variety. On tasks where such large and diverse datasets have been applied, monumental increases in both robustness and ability to generalize have been shown. Datasets for both object detection and instance segmentation lag behind this trend, traditionally requiring human annotations, no clear path to scaling these datasets to the size and diversity, shown to be so beneficial in other domains, has been identified.

This thesis examines an alternative approach to generating weakly supervised object detection and instance segmentation datasets. It differentiates itself from previous weakly supervised approaches by leveraging language modelling as part of its annotation pipeline to extract relevant information from image captions and thus achieve a greater variety of pseudo labels. This thesis presents the underlying approach to data generation in detail. By training a segmentation model on our synthetic data and through qualitative examinations and quantitative evaluations on benchmarks, the strengths, and weaknesses of this approach analyzed. Finally, an in-depth analysis of the limitations of our approach is presented.

Acknowledgements

I would like to extend my gratitude to Prof. Zhang and Damian Mrowca for giving me the opportunity to work on this exciting topic and their excellent guidance throughout this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Much of the recent advances in many domains of machine learning are in part fueled by the ability to harness large and varied datasets, as methods have made it possible to train on weakly-labeled or even unlabeled datasets.

Contrary to these recent advancements, object detection and instance segmentation have not been able to profit from this trend to the same degree. Unlike collecting image-text pairs or unlabeled text, there exists no simple method of collecting vast amounts of annotated images by scraping the internet. Traditionally, like most other fields in Machine Learning, the creation of object detection datasets required human annotation, making scaling beyond millions of images cost prohibitive, and as a consequence, the sizes and diversity of common detection and segmentation datasets now lag far behind that of other domains.

Much effort has been spent trying to bridge this gap, and allow training datasets to scale to the size and variability that has proven so crucial in other machine learning tasks.

This thesis aims to explore an alternative approach to scaling object detection and instance segmentation datasets, creating a weakly annotated dataset by automatically annotating images obtained from a web-scale dataset of image-text pairs. Compared to previous approaches, this approach leverages language modelling as part of its annotation pipeline to extract more information from an image's caption and generate more relevant pseudo labels.

1.1 Thesis structure

This thesis is organized into several chapters, each focusing on different aspects of the research undertaken:

1. **Chapter 2: Background**
   This chapter offers an overview of the existing literature in the domain of instance segmentation, synthetic training data generation, and model architectures. It identifies a gap in the current body of knowledge and sets the stage for the research questions addressed in this work.
2. **Chapter 3: Approach** chapter 3 first delves into the novel approach taken to generate a completely synthetic dataset. The subsequent sections describe how an existing segmentation model was modified to facilitate promptable instance segmentation and the approach taken to train it on this synthetic data.
3. **Chapter 4: Results**
   chapter 6 presents the model's application's on downstream tasks and its performance on 2 instance segmentation benchmarks to illustrate the effectiveness and drawbacks of this approach.
4. **Chapter 5: Limitations and Future Work**
   The implications of the results are discussed in this chapter. It interprets the findings in the context of the broader literature, offering insights and drawing connections. This concluding chapter summarizes the main findings, contributions to the field, and potential avenues for future research.
5. **Chapter 6:** chapter 6 summarizes the most important findings of this thesis.
6. **References**
   A list of all sources, papers, datasets, and tools cited or referred to throughout the thesis.
7. **Appendices**
   Supplementary materials, including tables and more detailed explanations not listed in the main body of this thesis.

Chapter 2

Background

This chapter introduces the reader to the tasks of object detection and instance segmentation, delves into the similarities of these 2 tasks, and presents the model architectures and datasets that have traditionally been used to solve these tasks.

In section 2.5, the previous approaches that have been taken to scale detection and segmentation datasets beyond what is possible with mere human annotations are presented and sets the stage for chapter 3, which delves into a weakness exhibited in many of the current approaches, which this thesis aims to address.

2.1 Object Detection and Instance Segmentation

2.1.1 Object Detection

Object detection, a central task in computer vision, is the task of identify and spatially locating objects within images. This differs from image classification, which provides content labels to the whole image, and image semantic segmentation, which offers pixel-wise categorization.

More precisely, given an image and one or more possible classes, the goal is to derive:

- **Bounding Box**: A rectangular enclosure, typically determined by the (x, y) coordinates of its upper-left vertex alongside its dimensions (width and height).
- **Class Label**: This label indicates the predicted category of the contained object.
- **Confidence Score**: A metric that gauges the model's certainty in its detection.

for each object in an image that can be described by one of the provided class labels.

2.1.2 Semantic Segmentation

Semantic segmentation is a computer vision task that involves classifying each pixel in an image into a predefined category. Instead of providing a bounding box around an object of interest (as in object detection), semantic segmentation provides a detailed understanding by assigning a class label to every pixel, highlighting even the minute details of an object or scene.

2.1.3 Instance Segmentation

Instance segmentation combines the principles of object detection and semantic segmentation. Unlike object detection, which identifies objects by bounding boxes, and semantic segmentation, which assigns pixel-level labels to object categories, instance segmentation goes a step further by not only detecting objects but also delineating the boundaries of each individual object instance with pixel-level accuracy. This task brings a finer level of detail and granularity to object understanding, enabling algorithms to discern and differentiate multiple instances of the same object class within an image.

More precisely, given an image and one or more possible classes, the goal of instance segmentation is to derive:

- **Mask**: A binary mask that covers the object in the image.

- **Class Label**: This label indicates the predicted category of the contained object.
- **Confidence Score**: A metric that gauges the model's certainty in its detection.

for each object in an image that can be described by one of the provided class labels.

Object detection and instance segmentation are roughly comprised of the same sub-tasks (object localization and classification). Because of this, both the model architectures used to solve these tasks and the datasets used to train and evaluate models are very similar.

2.1.4 Closed-Vocabulary segmentation and detection

Traditionally, object detection and instance segmentation models have been constructed to be closed-vocabulary, meaning they are designed to only be capable of detecting a pre-defined set of classes. Detecting or segmenting objects outside this set, or objects at a finer level of granularity (for example, detecting `'sports'` cars and `'vintage cars'`, if the model was only designed to detect the class `'car'`) is not possible without retraining at least part of the model.

2.1.5 Open-Vocabulary and Zero-Shot detection

Open-vocabulary object detection and segmentation refers to the task of identifying and localizing instances from categories that are not necessarily seen during the training phase, and can in theory be described by any natural language expression. Unlike traditional detection and segmentation, which operates within a closed set of predefined classes, open-vocabulary detection and segmentation operates in an open-world setting, allowing for the recognition of objects beyond a fixed vocabulary.

Zero-shot object detection (ZSD) and instance segmentation (ZSI) takes this further and imposes the restriction that the model must detect object classes not seen during training.

Both tasks are typically approached by training a model to output its prediction as an embedding of the class label, instead of just predicting per-class logits. During inference, this output is compared to the embeddings of the query.

2.2 Metrics and Evaluation

As with any other field in Machine Learning, object detection and Segmentation have dedicated methods to assess their accuracy, robustness, and generalization capabilities.

This short section covers the typical evaluation metrics that are reported and used to compare different models performance with each other.

2.2.1 Evaluation Metrics

1. **Intersection over Union (IoU):** IoU measures the overlap between the predicted bounding box (or mask) and the ground truth bounding box. It is computed as the ratio of the area of intersection to the area of union between the two boxes or masks. When evaluating a model's performance, a predicted box or mask is considered to match a ground truth instance, if its IoU with a ground truth instance exceeds a predefined threshold.
2. **TP, FP, FN** These describe how detections are categorized: True Positive: correctly identified instance. False Positive: Instance that is predicted by a model but does not correspond to any ground truth instance. False Negative: model fails to detect an instance.
3. **Precision and Recall:** Precision quantifies the ratio of correctly detected instances to the total instances predicted, while Recall measures what percentage of the instances in the image were correctly identified:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{2.1}$$

4. **Average Precision (AP):** AP is the average precision over all possible recall values (The area under the Precision-Recall curve).
5. **AP at Different Instance size** Object detection models are also evaluated at different instance size. Some architecture work better for detecting smaller objects, while others excel at detecting larger objects.
6. **mAP (mean Average Precision):** mAP aggregates the AP scores across all object categories. This is typically the metric on which models are ranked on leaderboards.

2.2.2 Cosine Similarity

The cosine similarity between two vectors A, B is the angle between them, and is defined as:

Definition 2.1 (Cosine Similarity)

$$CosineSimilarity = CS(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{2.2}$$

In the case that A and B are normalized, the Cosine Similarity is simply the dot product.

This metric is often used to gauge the similarity between embeddings produced by a model.

2.3 Techniques and Methodologies

The subsequent section describes the CLIP model which, though not a segmentation or detection model, plays a crucial role in this domain due to its ability to unify image and text embedding spaces.

The remaining subsections delve into 3 specific architectures that have been proposed to tackle the tasks of object detection and instance segmentation. As these two tasks are so similar, the architectures devised for them are often almost identical, and switching between an object detection model and instance segmentation model often only involves a modification of a final module, to output either a mask or a bounding box prediction and a change of learning objective. While numerous other architectures exist, they are not of primary relevance to this thesis.

2.3.1 CLIP [1]

Contrastive **L**anguage–**I**mage **P**re-training (CLIP) is a multimodal model consisting of a vision encoder and a text encoder, which encode images and text into a shared embedding space such that similar samples are close (high cosine similarity).

CLIP's training loss is a contrastive loss, rewarding the model from mapping associated image-text pairs to embeddings with high cosine similarity, while attaining low cosine similarity with all other samples in a batch.

This simple training objective and architecture allows CLIP style models to be pre-trained on vast datasets of weakly labeled image-text pairs (depending on the implementation, from 400M [1] up to 5B image-text pairs[2]) as these datasets can be easily sourced from the internet. CLIP models achieve strong zero-shot performance on a variety of image classification tasks and have been adapted to numerous downstream applications such as object detection [3], knowledge distillation [4], zero-shot text-to-image [5], image retrieval [6] and numerous other applications.

2.3.2 Region Based Convolutional Neural Networks

Mask-RCNN represents the culmination of a sequence of significant developments in the realm of detection and segmentation models, progressing from RCNN to Fast-RCNN, then Faster-RCNN, and finally to Mask-RCNN.

R-CNN [7]

The first incarnation of region based Convolutional Neural Networks, R-CNN, approaches the object detection task by using a Convolutional Neural Network to extract features from image crops. Given an input image, $\sim$

2k class agnostic region proposals are generated using the selective search algorithm (a non-ML algorithm designed to segment images with high recall, this algorithm is static and does not change during training).

The input image is then cropped to these bounding boxes and each of these image crops is then passed through the CNN to extract features of the cropped image, which are then used to classify the crop with multiple class-specific linear SVMs.

As R-CNN can and will produce many near-duplicate predictions, to collapse these to unique predictions, non-maximum suppression filters out detections with high overlap with higher scoring detections.

Fast R-CNN [8]

Although R-CNN brought a significant increase in detection accuracy, the approach was limited by the need for a separate forward pass of the CNN for each image crop.

Fast R-CNN follows the general approach of R-CNN using selective search to produce a set of region proposals for an input image, but improves on R-CNN by extracting image features with only a single forward pass of the CNN.

Instead of cropping the image prior to feature extraction, the resulting feature map is cropped to the region proposal bounding boxes. These crops are pooled to a uniform size (7×7 in [8]). These pooled feature maps are referred to as Regions of Interest (RoI). Fully connected layers map these RoIs to bounding box predictions and object classifications.

Faster R-CNN [8]

The improvement brought on by requiring only a single forward pass of the CNN shifted the computational bottleneck to region proposal generation. Faster R-CNN addresses this by utilizing a Region Proposal Network (RPN) to produce region proposals instead of selective search.

A CNN extracts features from an input image. By applying the RPN to a section of this feature map using a sliding window approach, multiple region proposals are predicted at each window. These predictions are made relative to anchor boxes (reference boxes with a predefined aspect-ratio and size).

Allowing the region proposal and detection mechanism to share the same CNN greatly reduces the computational cost, but makes training the network more involved.

Mask R-CNN [9]

Mask R-CNN extends Faster R-CNN by adding a convolutional head that predicts a binary mask in parallel to the bounding box and classification heads.

Limitations

This approach at the time was very successful at pushing the state of the art, it however relied on hand-designed components such as anchor-box generation and non-maximum suppression (and also selective search for pre Faster R-CNN approaches), their design and the chosen thresholds and constants would impact model performance. Further, region-based approaches generally struggle to detect small objects.

R-CNN approaches also generally view the task of object detection and instance segmentation as separate subtasks. With proposal generation, object classification and bounding box prediction decoupled, not allowing the model to be trained end-to-end.

2.3.3 DETR [10]

DETR (**De**tection **Tr**ansformer), introduced in the publication "End-to-End Object Detection with Transformers", is one of the first successful object-detection models based on a transformer.

DETR addresses some limitations inherent to Region based approaches, namely the need for hand-designed components, such as non-maximum suppression, anchor centers, anchor generation and region proposal generation as well as the inability to train such models end-to-end.

To circumvent the need for any prior knowledge and train a detector end-to-end, DETR abstracts the object detection task to a direct-set prediction task (given an image, predict the set of objects that appear in it) and solves the 3 subtasks (localization, classification, and confidence scoring) with a single unified architecture.

Architecture

DETR uses a 2 stage design, first a pretrained CNN extracts image features. A transformer-encoder encodes these features. The transformer decoder takes 91 learned object queries and applies alternating self-attention and cross-attention (learned object queries attend to encoded image features). This produces 91 decoder output tokens, which are mapped to bounding boxes and classification using a shared FeedForward Network (FFN). The transformer architecture allows the learned query embeddings to attend to the whole image, which improved accuracy on large objects.

Training

During training, a bipartite matching loss finds a 1:1 matching between predicted objects and ground truth objects, other predictions are matched with a `'No Object'` class, the matching is based on the IoU between predictions and ground truth boxes. This rewards the model for making only a single confident prediction per ground truth object, and thus negates the need for any non-maximum suppression to remove duplicate predictions.

Limitations

The original DETR implementation required $10\times$ more training epochs to converge than Faster R-CNN, and DETR's performance was even worse on smaller objects.

Both limitations were addressed in Deformable DETR [11], which uses sparse deformable attention in the transformer instead of dense attention to address the slow convergence and eytracts multiscale feature maps to better capture smaller objects.

2.3.4 SAM [12]

SAM (Segment Anything Model) is a prompt-able segmentation model. SAM is introduced alongside and trained on the SA-1B Dataset (described in subsection 2.5.5). Given an image and prompt(s), SAM predicts a single or multiple segmentation masks. These prompts can either be a point, a bounding box, a binary mask, or a text prompt. For each mask, SAM also predicts the IoU (Intersection Over Union) for each mask with the underlying object, which serves as a way to gauge the model's confidence in a mask.

Architecture

SAM's architecture loosely resembles DETR's design (subsection 2.3.3) by using a transformer to map learned embeddings and image features to mask predictions. DETR, however, is modified to allow prompt inputs and uses a large pretrained vision transformer instead of a CNN to extract image features.

A prompt encoder maps input prompts to prompt embeddings. These prompt embeddings are either sparse or dense, sparse embeddings result from point, bounding-box or text prompts, dense embeddings result from mask prompts. The sparse embeddings are concatenated with a set of learned input embeddings, the dense embeddings are added to the image embeddings.

SAM learns N+1 (N=3 for the released SAM models) query embeddings for the mask decoder in total, N of these embeddings each correspond to an output mask, the other embedding corresponds to the IoU prediction.

These learned embeddings are concatenated with the sparse prompt embeddings, which along with the image embeddings are mapped to mask and IoU predictions by the mask decoder. The mask decoder is a modified transformer decoder and features 2 layers, each layer performs self-attention, token to image attention, point-wise MLP on each token and image to token attention. Notably, the computation of image features can be done without any knowledge of the prompt, allowing for pre-computation.

Training

The backbone is initialized to the weights obtained in MAE [13], there are 3 versions of SAM corresponding to 3 different backbone and transformer sizes, ViT-B (91M parameters in total), ViT-L (308M Parameters in total) and ViT-H (636M parameters in total).

Training occurs interactively, which simulates the interactive use of a segmentation model. Initially, SAM is prompted randomly with either a bounding box or point prompt. In each subsequent step, given the previous output of the model, a point is randomly selected from the error region, this point alongside the previous mask output constitute the next prompt. In total 11 iterations are performed, 8 with randomly samples points and 2 with just the previously predicted mask as the prompt.

Applications

The approach of SAM, to facilitate the use of prompts, and not restricting it to only recognize a predefined set of categories, lends itself to numerous downstream applications with minimal modifications, similar to the way in which vision foundation models can be applied to downstream tasks that were not even considered during development. Some of these applications include:

1. **Object Proposal Generator:** Prompting SAM with a grid of evenly spaced points produces a set of proposal masks, by filtering based on confidence and stability you end up with a set of object proposals.
2. **Lightweight Segmentation Model:** SAM's mask decoder is small enough in terms of parameters that inference can be done quickly on a CPU. By precomputing the output of the much larger vision encoder, segmentation can be performed interactively without any dedicated hardware. making it ideal for web-based based applications.

Limitations:

Although SAM is a very capable segmentation model that provides strong results in a wide array of scenarios and lends itself to be used in many downstream tasks, it does have its limitations. SAM relies on a large Vision Transformer to extract image features, which is computationally expensive to perform, making the larger models unsuitable for real-time applications such as Augmented Reality. Further, SAM has been shown to exhibit subpar performance in some niche applications, such as segmenting medical images [14] and requires prior human knowledge to provide prompts that lead to good results.

2.4 Datasets and Benchmarks

As with any discipline in Machine Learning, Object-Detection, and Instance-Segmentation feature dedicated datasets and benchmarks that allow a model to be trained and its performance to be compared with other's. This section delves into some prominent datasets and benchmarks that have shaped this field and attempts to show the trends of the design of these datasets if following.

2.4.1 PASCAL VOC [15]

The PASCAL Visual Object Classes (VOC) challenge emerged early on as one of the most influential and widely recognized benchmarks for these tasks. Initiated in 2005 and lasting until 2012, the PASCAL VOC challenge provided a standardized platform for evaluating and comparing the capabilities of various computer vision algorithms across different object categories and scenarios.

The exact number of images and annotations varied from year to year. For the last installment of this challenge (2012), the PASCAL VOC benchmark encompassed 20 object categories over 11.530 annotated images, 27,450 bounding boxes and 6,929 segmentations. These categories include common objects such as `'cars, dogs, chairs, airplanes, and bicycles'`. Each image in the PASCAL VOC dataset is manually annotated with bounding box coordinates and object class labels. For instance-segmentation, pixel-level annotations indicating object masks are also provided.

The PASCAL VOC challenge ranks entries on their mean Average Precision (mAP) metric. This metric rewards both high precision (accurate detections) and high recall (capturing most instances) and provides a comprehensive assessment of an algorithm's capabilities across multiple object categories.

For instance, segmentation, the PASCAL VOC challenge introduced the mean Average Precision for Segmentation (mAP_{seg}). This metric extends the concept of object detection mAP to the instance segmentation task by evaluating the accuracy of both object localization and pixel-wise segmentation.

2.4.2 COCO [15]

COCO (**C**ommon **O**bject in **Co**ntext), introduced in 2014, is perhaps the most important object detection and segmentation dataset. It consists of 328k images, and 2.5M labelled instances across 80 distinct object categories, instances are provided both as bounding boxes and segmentation masks. COCO has on average 7.7 instances per image, surpassing the 2.3 of PASCAL VOC. The objects annotated in the images are objects in the 80 COCO classes, chosen to be "easily recognizable by a 4-year-old", these categories include categories such as `'Car'`, `'Cow'`, `'Airplane'`, and `'Chair'`. Further, for each image, multiple human generated captions are provided.

Unlike traditional datasets that often featured isolated objects against neutral backgrounds, COCO emphasizes contextual diversity. The dataset curators explicitly aim to gather a dataset with "non-Iconic" images where the object in question appears in its natural context. Images encompass scenes with multiple objects, complex backgrounds, occlusions, and objects at varying scales. This contextual richness mirrors the challenges posed by real-world scenarios, making COCO an ideal dataset for assessing the robustness and versatility of object detection algorithms.

Annotation Pipeline

COCO's images are sourced from Flickr by searching for pairwise combinations of object categories, this is done to avoid iconic images of single large centered objects. All COCO annotations are created by humans, with workers sourced on Amazon's Mechanical Turk. The authors report a cost of 22 worker hours per 1,000 segmentations, and a varying quality of segmentations between annotators.

RefCOCO [16]

The dataset's primary objective is to understand and localize objects based on natural language descriptions, bridging the gap between visual content and linguistic references.

RefCOCO is an extension of COCO, and is provided in 3 variations, RefCOCOg, RefCOCO and RefCOCO+.

These datasets are derived from COCO images, but, instead of annotating each instance with a single class label (for example: `'Giraffe'`, `'car'`), objects are annotated with natural language sentences that more precisely describe both the object and its location (`'an adult giraffe scratching its back with its horn'`, `'the sports car in a parking spot'`).

As such expressions are significantly more time-consuming to gather, all 3 datasets are considerably smaller than COCO, containing between 20K and

26K images with 50K to 55K annotated instances. The expressions describing the objects contain an average of 3.61 words for RefCOCO, 3.53 words for RefCOCO+ and 8.43 words for RefCOCOg. All annotations are created by humans. RefCOCOg was annotated using workers on Amazon's Mechanical Turk in 2 stages, in the first stage, a human created expressions for the objects in an image, in the second stage another human was provided the image and the expression and was tasked with finding the indicated object to validate the expressions' accuracy. RefCOCOg and RefCOCO+ were created with a similar 2 stage process but formulated as a game (ReferitGame).

Limitations

Though at the time, a detection dataset with over 300k images with 80 classes was an improvement over the state of the art, such a size is considered to be small by today's standards. Further, the average resolution of an image in COCO is 640×480 which also lags behind recent trends.

In [17] the authors analyze COCO to determine the racial and gender bias and find the dataset is heavily skewed towards white male individuals, with lighter skin individuals $7.5\times$ more common than darker-skinned individuals and male individuals $2\times$ more frequent than females.

2.4.3 Open Images Dataset V4 [18]

The Open Images dataset addresses the need for more extensive coverage and nuanced evaluation scenarios, enriching the understanding of object detection in complex, real-world contexts by enlarging the number of training images and training instances and providing more object classes.

Open Images consists of 9.2M images, 30.1M image-level labels. Only a subset of the datasets, consisting of 1.9M images, includes instance annotations, this encompasses:

- 15.4M bounding boxes for 600 classes
- 2.7M instance segmentations for 350 classes
- 66.3M point level annotations

The 15.4M bounding boxes means the dataset contains 15$\times$ more bounding boxes as the next biggest dataset (COCO). The resolution of the images is between 256×256 to 2048×2048.

2.4.4 Segmentation In The Wild [19]

SEGINW (**Seg**mentation **In** the **W**ild) is a benchmark to evaluate the Instance-Segmentation capabilities on a diverse set of scenarios. The benchmark consists of 25 datasets. For each dataset, a list of possible objects is provided, the goal is to identify and segment these objects. The mean-Average Precision (mAP) for each dataset is reported, submissions are ranked by their average score across all datasets.

Although SEGINW does provide training data, it is comparatively small, on average each dataset has only 500 training images, though this is unbalanced, many categories having less than 20 training instances in total. as some datasets feature object categories not present in any major detection datasets, for example ginger, garlic, tablets, and chickens. Good performance on SEGINW necessitates training on auxiliary data and the ability of the model to generalize to new unseen classes.

2.5 Scaling Segmentation Datasets beyond 2 Million Images

As machine learning (ML) has witnessed rapid advancements in recent years, there has been a discernible shift towards larger datasets. The importance of large datasets in the success of ML models can't be emphasized enough, and this benefit has been demonstrated in a variety of machine learning applications, for example:

1. **Image Classification:** According to [20] of the current top 20 most accurate models on the ImageNet1K benchmark (top-1 Accuracy), only 4 have less than 1 Billion model parameters (The remaining models have over 360M parameters) and all the top 10 models were trained on datasets with at least 3 billion images.
2. **Language Modelling:** Due to the ease of curating large text datasets by scouring the internet, these datasets have ballooned in size, with recent models such as Llama-70B being trained on 2 Trillion tokens.

There are multiple reasons why such large datasets are beneficial. In [21] the authors analyze contrastively pretrained Vision-Language models (such as CLIP) and determine the main contributor behind the unprecedented robustness exhibited by these models is the increased diversity exhibited when scaling dataset size. Increased robustness is only one of the benefits to scaling dataset size and variability:

- **Robustness:** Larger datasets tend to encompass a much wider distribution and variety of data points, aiding models in understanding nuanced patterns and edge cases.
- **Generalization:** With ample data, models are less likely to overfit to a narrow subset and can thus generalize better to unseen data.
- **Representation:** Expansive datasets offer a more holistic representation of the real-world scenario, making predictions more accurate and reliable.

The previous section section 2.4 presented the traditional approach to curating detection and segmentation datasets; images are annotated by hand, with images also curated by manually searching for favorable images. This methodology lies in stark contrast to the recent trends of scaling dataset size. To date, no dedicated dataset in this domain surpasses the 2M image threshold. Equally important, the breadth of data variability in these datasets is limited. Annotations predominantly encapsulate instances in a constrained set of classes, rarely exceeding 1,000 distinct categories. This results in a lack of granular information within annotations, with annotations only describing broad class affiliations such as `'car'` rather than more descriptive labels such as `'vintage red sports car'`. This diversity of annotations is an aspect shown to be key to the robustness of contrastively pretrained Vision-Language models. An exception to this rule can be found in the refCOCO datasets; they, however, contain fewer than 26,000 images. The lack of such labels limits the ability to train models which can be queried using natural free-form text.

In order to benefit from the gains in performance other fields of Machine Learning have seen by increasing dataset sizes, detection and segmentation datasets need to increase not only the number of images and annotations, but also the number and variability of annotations.

As demonstrated in the cost calculation in [15], scaling a hand-annotated segmentation dataset beyond a few million images is prohibitive in terms of both cost and time (Using the cost estimate of 22 hours per 1000 masks and assuming an average image requires 7.7 annotations like COCO, annotating 1B images by hand would require $\sim$169M hours of work, which exceeds some estimations of the work required to build an Egyptian pyramid [22]).

Clearly, to scale the size and diversity of detection and segmentation datasets beyond their current state, an approach other than naive human annotation must be taken.

Unfortunately, unlike image-text datasets, which can be collected easily by scouring the internet for images and associated texts, such a simple solution does not exist for segmentation and detection datasets. However, such large datasets of image-text pairs can serve as a basis to derive detection datasets, and multiple approaches have been successfully applied to achieve this:

2.5.1 DETIC [23]

Detector with **I**mage **C**lasses attempts to train an object detection model capable of detecting a much wider variety of classes than previously possible by using a weakly-supervised approach. To bridge the gap made by the lack of detection datasets with such a diversity of annotations, the authors propose a novel training approach, training jointly on detection and classification datasets, allowing the classification module of an object detection model to learn from the much larger and more diverse image-text datasets available.

Architecture

The DETIC model architecture is modified from Faster R-CNN to output CLIP aligned language embeddings instead of per-class logits for each prediction.

Training Data

DETIC relies on the insight that existing approaches to training localization networks already work very well at detecting unseen classes, and what is needed to expand beyond the existing classes is to build a stronger classification head.

DETIC is trained jointly on 2 data sources, conventional detection data and image-text supervised data. The smaller detection datasets (COCO [15] (328k images, 80 classes), LVIS [24] (164k images, 1230 classes)) are combined with much larger image-level supervised datasets (ImageNet21k [25] (14M images 21,843 classes) and Conceptual Captions [26] (3.3M images each with a human generated free form captions)).

To train jointly on detection and classification data, the task of object detection is logically split into two parts, object localization and object classification. Detection data is used to supervise both object localization and classification, while image-text pairs are used to supervise only the object classification (for such samples, the bounding box is assumed to be the biggest bounding box generated by the region proposal network). This training recipe effectively allows the model to learn the semantic embeddings of the 21k categories in ImageNet, as well as the human generated natural language captions of CCM. Further, as DETIC learns to predict a text embedding instead of per-class logits, DETIC can function as an open-vocabulary detection model, by encoding object queries using CLIP's text encoder and comparing their Cosine Similarity to the model's predictions.

The model achieved (at the time) state-of-the-art performance on a variety of benchmarks and showed a strong ability to transfer performance from one dataset to another. DETIC also exhibited strong performance in its zero-shot capability, the ability to detect classes not seen during training.

2.5.2 ViLD [4]

Vision and **L**anguage knowledge **D**istillation is a training method to train open-vocabulary object detection models by distilling knowledge from a pretrained teacher model (open-vocabulary image classification model) into a student model (a Faster-RCNN style 2 stage detection model).

Architecture

Similarly to DETIC, ViLD is a modification of the standard 2-stage region based CNN design of Faster R-CNN, again modified to output a classification as a language embedding. This output is aligned either with CLIP or ALIGN, depend on which is used as the teacher model.

Training

Knowledge Distillation is performed in 2 ways. ViLD is trained to align its classification embedding with the teacher model's vision encoding of a cropped region and the teacher model's text encoding of the instance label. The alignment, with the image crop embedding, does not require the crop's class affiliation to be known during training, thus allowing ViLD to learn object classes that were never annotated.

Limitations

The ViLD approach relies on the assumption that the embedding of an image crop aligns with the embedding of its associated label. RegionCLIP [27] perform an experiment in which a detection dataset (LVIS) is cropped to its ground truth bounding boxes and CLIP model is then used to classify the crops. The CLIP model achieved an accuracy of only 19.1 %, compared to an accuracy of 59.6 % on ImageNet1k, which features a similar number of classes. This finding indicates that there is a gap between the clip text encoding of an instance label and the CLIP vision encoding of an associated image crop. Training an object-detection model using such generated embeddings may not be ideal, especially when the associated text embeddings will be used to classify object proposals.

2.5.3 3Ways [28]

"Three ways to improve feature alignment for open vocabulary detection" (3Ways) explore training open-vocabulary object detection models with better alignment between visual and text features. The paper covers 3 ways of doing this (hence the name), the third step of the approach is particularly interesting as it employs a self-training approach to annotate existing image-text pairs to produce a larger, more diverse dataset, on which the model is then retrained.

Self-Training

In the first 2 steps (2Ways) the authors describe how they train a strong open vocabulary object detection model using existing datasets. This model is then used to derive pseudo labels for an dataset of image-text pairs scraped from the internet (Conceptual Captions: 12M images). For a given image, the whole image caption is used to query the 2Ways model, the bounding box with the highest confidence is then used as the pseudo bounding box and the image caption the pseudo label. These resulting pseudo-labels are then used to retrain the same model, now on a much larger dataset, and resulted in a performance gain on the evaluated benchmarks.

Limitations

This approach relies on the assumption that the whole image caption is an adequate descriptor of (one of) the objects visible in the image. The dataset from which image-text pairs are sourced, Conceptual Captions [26] consists of human written annotations. Some examples taken from the publication are: "`actors attend the premiere at festival.`", "`side view of an aircraft on approach to land with landing gear down`", "`sculptures by person adorn trees outside the derelict offices`"[1].

These captions are typical of human derived image captions, where the caption describes the image as a whole, but does not explicitly mention any or all of the visible objects, meaning the derived pseudo label may be only weakly associated with the objects visible in the image. Examples of where such an approach suffers will be shown in section 3.1

[1] captions taken from CCM [26]

2.5.4 OWL-ST [3]

Open World Localization Self Training (OWL-ST) extends a previous method of producing open-vocabulary object detection models, OWL-ViT, and applies a similar approach to 3Ways, using this model to derive pseudo annotations for a web scale dataset ofimage-text pairs.

Architecture

The OWL-ViT model consists of a contrastively pretrained image and text encoder (taken from CLIP), the token pooling layer of the image encoder is removed, this produces a series of tokens instead of a vector embedding. These output tokens of the vision transformer are mapped to class predictions and box predictions using a linear projection and MLP, respectively. The class embeddings are aligned with the CLIP text-encoder encoding of the associated label.

Data Generation

To derive pseudo labels, OWL-ST takes a similar approach to 3Ways, using an existing model to generate them, in this case an existing OWL-ViT model. The OWL-ST approach differs in the queries used to query the detection model. In 3Ways, the query is simply the whole image caption, in OWL-ST, the caption is split into its constituent words and used to construct n-grams (up to length 10). Further, 3Ways only produces a single pseudo bounding box per image, OWL-ST keeps all predicted bounding boxes above a certain threshold.

Limitations

The handling of captions can lead to a similar limitation described above for 3Ways, namely that captions typically describe an image as a whole and leave out information that would be obvious to a human. For example, to a human reading it, the caption "`young woman trekking through the forest`"[2] does not need to explicitly mention that a tree is visible in the image as this is obvious to anyone who has ever seen a forest.

But it should be mentioned that the authors explicitly avoid using any techniques to augment the set of candidate queries beyond the n-grams to avoid any bias that would impact the diversity of queries.

[2] caption taken from CCM [26]

2.5.5 SA-1B [12]

Introduced alongside the SAM model, SA-1B is a segmentation dataset consists of 11M high-resolution images with 1.1B masks. Contrary to previous segmentation datasets, 99.1% of SA-1B's masks are entirely machine-generated. SA-1B's masks are not annotated with semantic labels and have no inherent semantic 'meaning' (i.e a mask does not necessarily correspond to a 'thing' with a clear label). SA-1B stands out in the landscape of segmentation datasets because of its high resolution, with images having a resolution of 3300×4950 pixels on average.

SA-1B is particularly significant in the way it was curated. SA-1B demonstrates a novel method of making a single human annotation go a lot farther than naive approaches such as that used in COCO. The dataset is created with human interaction in multiple steps. In each stage, the model generates a new dataset, which is then improved by humans and used to retrain itself. Each stage results in a larger dataset and a stronger model. This is done until the model is capable of creating a dataset with no human supervision.

The dataset creation pipeline is split into 3 steps:

1. **Assisted Manual stage:** An initial SAM model is trained on publicly available segmentation-datasets. This model is then used to assist Professional annotators to annotate a subset of the 11M images interactively. In this stage, 120k images with 4.3M masks were collected. SAM is iteratively retrained as more data becomes available. As the segmentation is done interactively, requiring the human to only apply point queries to include or not include a region, the annotation is $6.5\times$ faster per image than the fully manual COCO procedure.
2. **Semi-automatic stage:** In this stage, images are segmented automatically and presented to annotators, who then annotate any further segments. The masks are generated by prompting SAM with bounding boxes produced by a bounding box detector (trained on data produced in stage 1). This stage produced 5.9M more masks on 180k images. In this stage, SAM is retrained 5 times as more data was collected.
3. **Fully automatic stage:** At this stage, enough annotations have been collected to train a robust model. This model is then used to annotate 11M images by being prompted with a 32×32 grid of points, only masks with high confidence are kept and then subsequently filtered by applying non-maximum suppression to remove duplicates. This results in 1.1B masks in total. These masks are not validated or improved by an annotator.

Although the mask generation of SA-1B is not fully autonomous and still requires humans to annotate data, it introduces a novel way of making

human annotations go much further. Only 0.9% of the masks were generated with any human interaction, while maintaining high-quality segmentations.

Chapter 3

Approach

Starting with section 3.1, I first outline a potential weakness of the previously described methods of scaling detection dataset size. The following chapters then describe a method of circumventing this weakness by generating object proposals based on an image's caption, and detail how these pseudo-labels are generated. In the following sections, a model architecture is described which is then trained on the generated data.

3.1 Research Gap

In section 2.5, I covered the previous approaches taken in the goal of scaling datasets for detection and segmentation beyond what is feasible with naive human annotations. These approaches let detection datasets in theory scale to billions of images, limited in size only by the size of image-text datasets.

This thesis aims to address a potential weakness present in many of these approaches, namely those that attempt to derive pseudo annotations from an image's caption.

This issue is best illustrated by an example; consider the image with an associated caption: "`A busy intersection in a city on a sunny day`"[1]. A human reading this caption, without seeing the image, will likely predict that this image contains:

- `Car(s)`
- `Motorcycle(s)`
- `Person(s)`
- `Traffic Light(s)`
- `Building(s)`
- `Road(s)`

The caption describes the image perfectly well, but barely mentions any of the constituent objects.

If this image-text pair serves as a training sample for DETIC, during training, the largest predicted bounding box is associated with the label "`A busy intersection in a city on a sunny day`"[2]. None of the objects that constitute the image are treated separately.

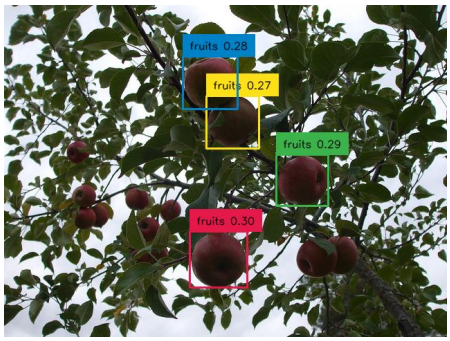
[1]Caption taken from COCO captions [29]
[2]Caption taken from COCO captions [29]

If this image-text pair serves as a training sample in the 3Ways approach, the image is queried for the most confident object corresponding to the whole caption. Figure 3.1a shows the prediction made when the detector is queried in such a way. The pseudo bounding box predicted encompasses almost the whole image, and again, none of the constituent objects are treated separately.

If this image-text pair shows up in OWL-ST training, pseudo labels are generated by querying this image with n-grams constructed from the words [`'A'`, `'busy'`, `'intersection'`, `'in'`, `'city'`, `'on'`, `'sunny'`, `'day'`]. Figure 3.1b shows the outcome of querying the image with these n-grams. Again, the detector matches the general label `'a busy intersection'` to almost the entirety of the image, as this is a fitting description of the whole scene, but misses any of the objects that actually constitute the image, such as the `'car'` and `'motorcycle'`.

(a) Pseudo label using whole caption

(b) Pseudo labels using n-grams



(c) Pseudo labels using n-grams

Figure 3.1: Example pseudo labels using previous approaches[3]

[3]Note, the detector used for these examples is Grounding-DINO and is thus not representative of what 3Ways or OWL-ST data looks like and serves mainly to illustrate the argument.

Figure 3.1c shows an example of where the OWL-ST approach is more suitable, given the caption "`A bunch of fruits hang on a branch`"[4], this method derives pseudo labels for the individual fruits, as this word is explicitly mentioned, but still assigns the label `'fruits'` to the apples.

The underlying issue is that these approaches attempt to derive pseudo labels from an image's caption by either assuming the caption describes a single object in the image or that the constituent words of the caption accurately describe objects in the image. This, however, is often not how images are captioned, the caption describes the image as a whole and leaves information that would be redundant for a human.

In the case that the image caption does not explicitly mention the objects visible in the image, these approaches lead to pseudo labels that are either only loosely related to the labeled instance, or no pseudo labels at all for objects not explicitly mentioned.

This issue becomes particularly pronounced when an object appears in its natural context, as then the likelihood of it being explicitly mentioned decreases, as it would be obvious to a human reading the caption (An annotation is more likely to mention a `'car'` if it is stuck in a river, than a car that is driving on a highway). As COCO [15] showed, these are precisely the types of situations in which these objects should be labelled, as this reflects the real world.

[4]Caption taken from COCO captions [29]

This issue is systematic to any method deriving proposals solely from the image caption. This issue is already present in human labeled datasets such as the captions taken from COCO presented above, but grows in relevance as datasets scale to web-scale weakly supervised datasets. In this setting, captions are only weakly related to the image. The first 5 captions of the LAION-5B dataset demonstrates this[5]:

- `"Weird Al Finally Gets First No.  1 Album with <i>Mandatory Fun</i>"`
- `"2020 Can-Am Outlander XT 850 in Ponderay, Idaho - Photo 1"`
- `"Kay Coughlan and Rosaline Higgins pictured at Bressies Charity lunch in aid of Larcc Cancer Support in Brasserie 15 Restaurant in Castleknock"`
- `"corporate stock certificate template 21 stock certificate templates free sle exle"`
- `"iHorse Racing 2 - PVP horse racing manager"`

What is required to circumvent this issue, is a method for going beyond only the words in a caption and instead using the caption to derive further candidates for objects that likely appear in an image but do not necessarily appear in its caption.

In OWL-ST, the authors explicitly avoid any methods of producing object proposals that go beyond only the words in the caption, so as not to introduce any bias inherent in these methods. But only focusing on words in the caption introduces its own bias, namely that many objects will be missed as they are rarely mentioned, especially if they appear in a scenario for which their appearance is obvious to a human (For example: a `'wheel'` in an image of a car, `'Window'` in an image of a house or `'leaf'` in an image of a plant). Using only labeled instances of these objects when they are explicitly mentioned in the caption, will bias their occurrence in the dataset to images in which they are the focus of the image (iconic images) or outside their natural context and therefore worth mentioning in the caption.

The approach presented in the following sections explores a method of bridging this gap by leveraging Large Language Models (LLM). The ability of LLMs to accurately model language is exploited to produce object proposals beyond the words in an image's caption.

[5]Captions taken from LAION 400M URL list found on [30]

3.2 Data Generation

3.2.1 Preliminaries

The approach used to generate a weakly labeled dataset presented here broadly follows 3Ways and OWL-ST in the sense that an existing open-vocabulary object detection model is used to generate pseudo-labels for images sourced from an existing image-text dataset. The approach diverges primarily in the method in which proposals for pseudo-labels are generated.

Figure 3.2 gives an overview of the approach; images and their associated texts are sourced from a Web-Scale dataset of image-text pairs. A Large Language Model (LLM) is prompted with the image's caption to produce candidate labels for objects that my appear in the image. These candidates are then used to query an open vocabulary object detection model to produce pseudo-bounding boxes, which are later used to train an open vocabulary instance segmentation model.

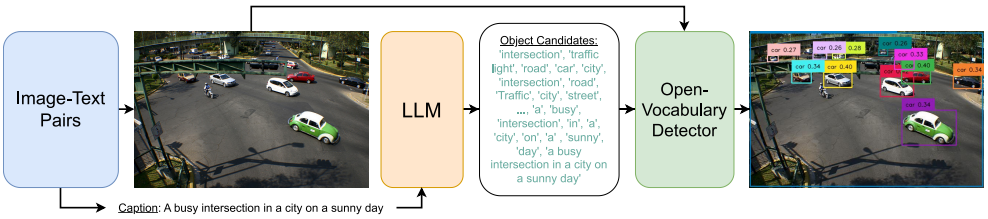


Figure 3.2: Overview of data generation method.

Such an approach will undoubtedly increase the variety of labels derived for each image and allow previously unlabeled instances to be assigned labels. The approach, however, has its own disadvantages, at each stage, the underlying model will impart its own weaknesses and biases onto the generated dataset. The language model will be limited in its ability to propose adequate candidates, both due to the inherent bias of the model and due to the prompts used to extract this information, giving an upper bound to the distribution of labels generated. The limitations of the detector will compound this, as no detector exhibits the ability to accurately detect objects across the full spectrum of natural language, and will impart its own bias when assigning queries to objects. A deeper dive into the individual limitations of each stage of this pipeline is presented in chapter 5.

3.2.2 Data Source

The image-text pairs are sourced from the High Resolution subset of LAION [31], a dataset consisting of 400 million image-text pairs with a resolution of at least 1024×1024 pixels. For copyright reasons, examples shown in this thesis are instead from COCO[15].

3.2.3 Extracting Object Candidates

Given an image-text pair, $(\mathbf{I}, t)$, we first want to derive as set of object candidates $\{o_1, o_2, ..., o_n\}$ from t, these candidates should describe objects that may appear in the image.

Previous approaches, either achieve this by using a single candidate, the whole caption, or constructing n-grams out of the words in the text. In this approach, the task of predicting object candidates is abstracted as the task of predicting the end of an adequate sentence, a task in which LLMs excel.

Choosing Prompt Templates

Generating multiple datasets, each with differing prompt templates to perform ablation studies is infeasible at this scale, the prompt should therefore be chosen before data is generated.

The initial plan was to choose a single good prompt, but it became apparent that different prompts work better in certain scenarios. This likely arises because the degree to which captions are related to the visible object varies.

Some prompts excel when the candidate is very intimately related to one of the words/sentences in the caption, but needs to only be altered slightly. For example, deriving `'brush'` from "`a child brushing her hair in the mirror`"[6] or `'surfboard'`" from "`A surfer riding a wave in the ocean`"[7].

Other prompts work well when the LLM needs to be more 'creative', for example deriving that there is likely a visible `'car'` if the image is captioned with `a stop sign with a sky in the background`"[8] or predicting `'knife'` from the caption "`Cutting board with various fruits, utensils, and spices`"[9]

Therefore, it was decided to use multiple (5) prompts and aggregate all proposals. These prompts are:

- `An image is annotated with '<Caption>' an object it may contain is:`
- `An image is annotated with '<CAPTION>' a thing it may contain is:`
- `An image that is described by '<CAPTION>' likely features a:`
- `If an image is annotated with '<CAPTION>' you might see a:`
- `If an image is described by '<CAPTION>' you might see a:`

The design of prompts has likely the greatest impact in the overall quality of data generation as the effects are compounded in subsequent stages. Limitations and alternative approaches are discussed in section 5.5.

[6] Caption taken from COCO captions [29]
[7] Caption taken from COCO captions [29]
[8] Caption taken from COCO captions [29]
[9] Caption taken from COCO captions [29]

Language Model

The Language Model used is the Bloom-7B1 model from BigScience [32, 33]. Despite the existence of larger and more performant Large Language Models, this 7B parameter model was chosen primarily for convenience, as it was small enough to fit on a single 24Gb GPU, which are plentifully available on the ETH university compute cluster[34]. Furthermore, this language model supports vllm [35] for inference, making it possible to handle millions of queries in an acceptable time frame.

Although this setup produced a reasonable quality in object proposals, switching to a larger, more performant model such as Llama-70B-2 gives stronger results, in the scope of this thesis however, using such large models was highly impractical as larger GPUs are not readily obtainable through the university's cluster, making inference on tens of millions of queries infeasible.

The language model is queried using a beam search, with 8 beams used, the best of 4 being returned as candidates. This results in 20 object proposals.

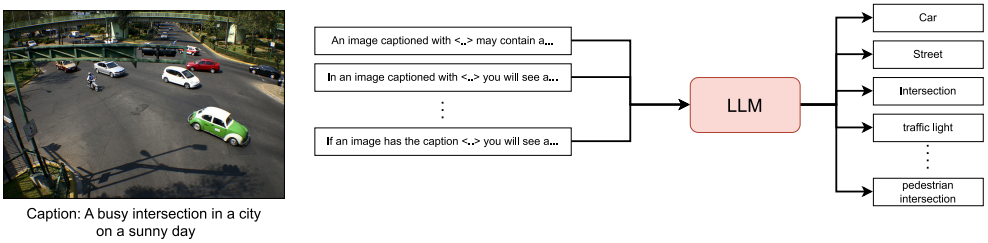


Figure 3.3: Overview of proposal extraction methodology

Increasing object proposals

In addition to the predictions made by the LLM, the whole caption t and all the words that constitute t are also added to the set of object candidates $\mathcal{O}$. This effectively makes the candidates derived here a superset of the candidates in the 3Ways approach, but, because no n-grams are added to the object candidates, not a superset of the OWL-ST approach.

3.2.4 Generating Pseudo Bounding Boxes

At this point, we now have a set of images and object candidates for each image $\{(\mathbf{I}, \{o_j\}_{j=1}^{N_i})_i\}_{i=1}^{N}$. Using an open-vocabulary object detection model, the image is queried for these candidates.

The open vocabulary object detection model used is Grounding-DINO [36], chosen as it is the strongest publicly available open-vocabulary detection model, that has shown the ability to generalize well to unseen categories [37], making it ideal for this use-case.

If a candidate matches an object visible in the image (the detector assigns it a score above 0.23 [10]), then the detected bounding box $\mathbf{b}$ and the CLIP text encoder embedding $\mathbf{c}$ of the corresponding object query o_i are kept as a pseudo label. The CLIP embeddings are computed using the ViT-L/14 model of the open-CLIP implementation [2]. The CLIP embeddings have a dimension of 768.

When this has been done for all images, this results in the dataset $\mathcal{D}^{semantic} = \{(\mathbf{I}, \{(\mathbf{b}_j, o_j, \mathbf{c}_j)\}_{j=1}^{N_i}\}_{i=1}^{|\mathcal{D}semantic|}$.

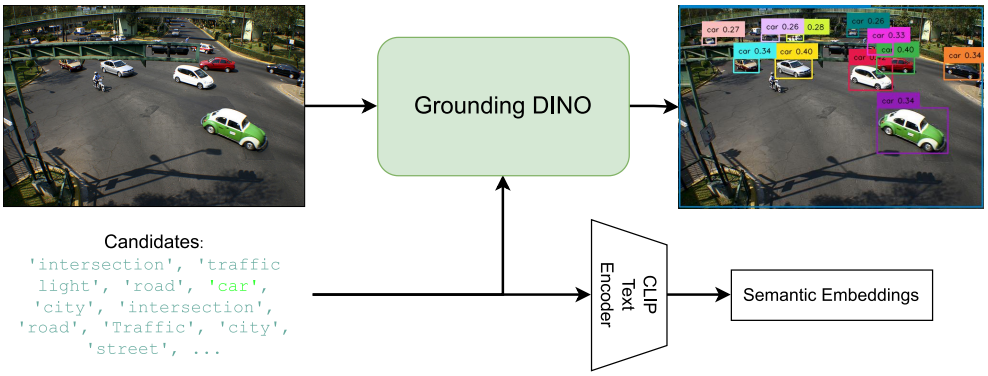


Figure 3.4: Generating Pseudo Bounding Boxes

3.2.5 Dataset Statistics

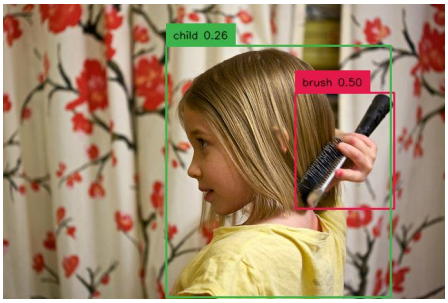
The data generation resulted in a dataset of 3.9M images with 27.8M bounding boxes. On average, just over 7 bounding boxes were produced for each image. All images have a resolution of at least 1024×1024

Filtering the images (by failure to download, NSFW content, Language, and images where no bounding box was generated), resulted in only $\sim$ 34 % of the images listed in the LAION URLs making it into the final dataset.

[10] Empirically found to work best

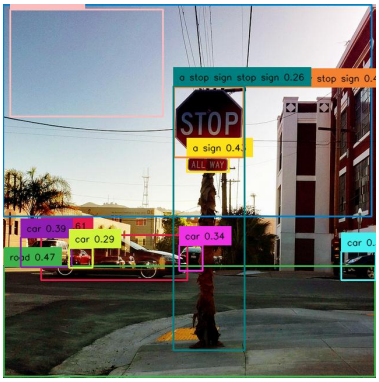
3.2.6 Examples

Figure 3.5 presents various examples generated by the data pipeline described above. The caption below the image is the caption used to derive object proposals. Note the ability to capture objects not explicitly mentioned in the caption, such as `'Brush'` in Figure 3.5a and `'Car'` in 3.5b and Figure 3.5c. [11]

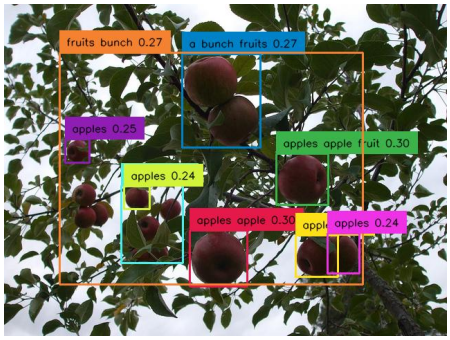


(a) "a child is brushing her hair in the mirror"

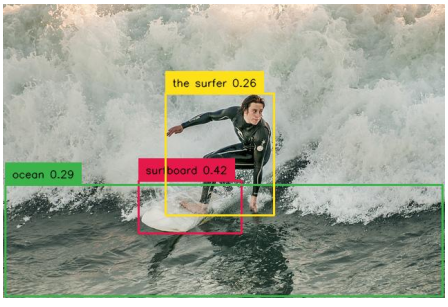
(b) "A busy intersection in a city on a sunny day"



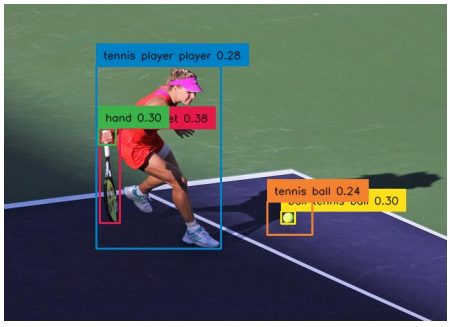
(c) "a stop sign with a sky in the background"



(d) "A bunch of fruits hang on a branch."



(e) "A surfer riding a wave in the ocean"



(f) "a woman running up to a tennis ball to hit it"

Figure 3.5: Examples of generated pseudo labels for COCO images.

[11] All captions taken from COCO captions [29]

3.3 Semantic-SAM

3.3.1 Why SAM

To analyse the effectiveness of this data generation method presented in section 3.2, naturally a model should be trained on it. There are many directions that can be taken at this stage. Recent progresses in approaches with early image-text fusion, such as Grounding-DINO, have shown strong results in open-set object detection. These approaches however limit downstream applications by requiring text queries to be known throughout the entire computation, limiting their use on retrieval and interactive tasks, where the computationally demanding steps can be pre-computed and shared among queries.

Models such as SAM are more suited for such applications, the 2 stage design of a large vision encoder coupled with a lightweight mask-decoder offers more flexibility in downstream applications. We chose SAM as a starting point for this. SAM is also already capable of producing high quality segmentation masks across a wide variety of scenarios. This allows us to piggyback on this ability and focus mainly on introducing semantic understanding while only preserving segmentation ability.

The remainder of this section describes the segmentation model that was developed as an extension of SAM within the scope of this thesis. This model will henceforth be referred to as Semantic-SAM. Semantic-SAM's architecture is a modification of SAM such that in addition to mask and IoU predictions, it also predicts:

- **Semantic Embedding:** For each predicted mask, output as a 768-dimensional vector.
- **CS Prediction:** Similar to SAM's IoU prediction, the model learns to predict the Cosine Similarity (CS) between prediction and ground truth. This was added to mirror the IoU prediction output of the original SAM and helps to filter out low-confidence predictions in downstream tasks.

An overview of the overall architecture is presented in Figure 3.6; A large vision transformer encodes an image into a series of tokens. A comparatively smaller prompt encoder encodes prompts. A mask decoder maps these image embeddings and encoded prompts to mask predictions. Semantic-SAM makes the same number of mask predictions as SAM, namely 3.

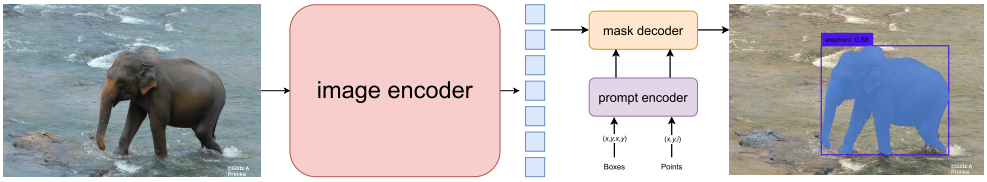


Figure 3.6: Semantic SAM Architecture overview.

3.3.2 Backbone

The vision backbone used for feature extraction is kept identical to SAM, which in turn was taken from [13]. The ViT image encoder operates on an input resolution of 1024×1024 and produces $64 \times 64 \times 256$ feature embeddings. Images are prepared by scaling the longest side to 1024 pixels, then padding the image to 1024×1024 and normalizing.

In subsequent steps, the $64 \times 64 \times 256$ image embeddings are treated as 4096 256-dimensional vectors.

3.3.3 Prompt Encoder

A prompt can be either:

- A point, represented with 3 numbers (x, y, i) (coordinates and a binary label for foreground/background)
- A bounding box, represented with 4 numbers $(x_{tl}, y_{tr}, x_{bl}, y_{br})$. Representing the top-left and bottom-right corners.

The original implementation also included functionality to handle textual prompts (input as CLIP embeddings) and mask prompts. The textual prompt functionality was not publicly released, and the mask prompt were not necessary for this thesis, both prompt types were therefore not included in Semantic-SAM. The encoding of point and box prompts is kept the same as in the original SAM implementation.

Points are encoded by first shifting to the center of the pixel (by adding 0.5), then its positional encoding is added to a learned embedding, there is a learned embedding for the top-left corner and one for the bottom right corner.

3.3.4 Mask Decoder

The main modifications were made to the mask decoder. Figure 3.7 shows the original SAM mask decoder architecture taken from Figure 14 in [12], and the modifications made to produce Semantic-SAM. The modifications allow Semantic-SAM to predict per-mask semantic embeddings in parallel to mask predictions.

Modifications are made before and after the transformer, with the transformer kept identical to the original SAM implementation.

In SAM, there are N+1 (N is the number of masks, in SAM and Semantic-SAM: N = 3) learned embeddings, N embeddings correspond to the mask predictions while one corresponds to an IoU prediction. Semantic-SAM doubles this to 2N+2 learned embeddings. The new learned embeddings correspond to the N predicted semantics and a CS score predictions.

The output tokens of the transformer that correspond to the original SAM implementation are treated identically to SAM. The N+1 new tokens are treated similarly. The N tokens corresponding to semantic predictions are mapped to 768-dimenstional vectors by N separate MLPs. The CS score token is mapped to N CS scores, also by an MLP.

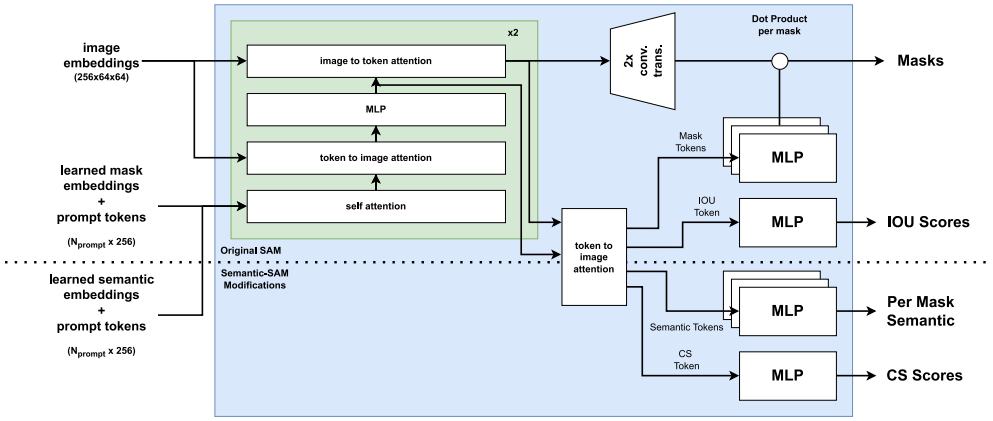


Figure 3.7: Semantic-SAM mask decoder

3.4 Training

3.4.1 Preliminaries

As SAM already exhibits excellent segmentation abilities, the model should be trained in a way that maintains this ability while training its semantic ability. To achieve this, all weights that can be initialized to SAM weights (the backbone, prompt encoder, mask decoder, learned embeddings for masks and IoU) are initialized to the public SAM weights at the onset of training. The semantic and CS prediction MLPs and the semantic learned embeddings are randomly initialized.

3.4.2 Training Data

In a first approach, Semantic-SAM was trained on only $\mathcal{D}^{semantic}$ with ground truth masks produced by prompting the public ViT-H SAM model with pseudo bounding boxes generated in the fashion described above. This training procedure, however, led to a visible decrease in the model's mask quality. It should be noted here that such an outcome is not surprising, as even in the original SAM implementation, the authors trained a special separate model to generate SA-1B and did not use the public models. This model was trained only on data at least partially human annotated, and is not publicly available.

To maintain SAM's high-quality mask output, subsequent training occurred jointly on $\mathcal{D}^{semantic}$ and SA-1B data. Within a single batch, these datasets were mixed at a ration of 4:1 (this ratio was the largest ratio that did not exhibit a visible decrease in the mask quality, mask quality comparisons with the original SAM are presented in subsection 4.1.1).

For the rest of the chapter, let $\mathcal{D}^{seg} = \{(\mathbf{I}, \{m_k\}_{j=1}^{N_i})_i\}_{i=1}^{|\mathcal{D}^{seg}|}$ refer to unmodified SA-1B data consisting of Images and associated segmentation masks without any semantic annotations.

3.4.3 Model Input

A training batch consists of 4 samples from $\mathcal{D}^{semantic}$, and 1 sample from $\mathcal{D}^{seg}$.

For each sample in $\mathcal{D}^{semantic}$, 2 instances are randomly selected. For each instance, a prompt is constructed using only its bounding box. Similar to the training of SAM, bounding boxes for both $\mathcal{D}^{semantic}$ and $\mathcal{D}^{seg}$ are augmented by adding random noise to the 4 coordinates to help the models' inference performance. The noise added is sampled randomly with a standard deviation equal to 10% of the box's side length, up to 20 pixels.

For samples in $\mathcal{D}^{seg}$ 4 ground truth masks are randomly selected, for every ground truth mask, a prompt is constructed by providing either a single point, multiple points, a bounding box or a mixture of these. Points are sampled at random from the ground truth mask. No background points are used so as not to complicate training.

3.4.4 Losses

The model's outputs, which for each image-prompt pair consists of 3 predicted masks, 3 predicted semantic embeddings, 3 IoU predictions and 3 CS predictions, are then used to compute the loss. The loss calculation differs between $\mathcal{D}^{semantic}$ and $\mathcal{D}^{seg}$:

$\mathcal{D}^{seg}$

For Segmentation data we have no semantic ground truths and these samples serve only to preserve the segmentation ability. The semantic outputs are therefore discarded, and only the predicted masks and IoU scores contribute to the loss. The loss is computed similarly to [12]. The sum of the focal loss and dice loss of the predicted masks and ground truth masks at a ration of 20:1 is used to supervise mask prediction, while the mean-square-error between predicted IoU and actual IoU supervises the IoU prediction branch.

$$L_{mask} = 20 \cdot L_{focal} + L_{sigmoid} + L_{iou\_MSE} \tag{3.1}$$

$\mathcal{D}^{semantic}$

For $\mathcal{D}^{semantic}$ data, only the semantic output (predicted text embedding and CS prediction) of the model contributes to the loss. The loss is calculated as the sum of a contrastive loss for the predicted semantic embeddings and the mean-squared-error between the actual CS and the predicted CS. This loss is summed at a 2:1 ratio.

$$L_{semantic} = 2 \cdot L_{contrastive} + L_{cosine\_similarity\_MSE} \tag{3.2}$$

Previous works have seen a significant gain in performance when using batch negatives [28] in a similar setting. Exactly how to apply this in this approach is not immediately clear. Instead, such a contrastive loss was applied to achieve a similar effect. The contrastive loss is calculated globally across a batch (across all nodes), this loss rewards the model for making semantic predictions that have a high overlap (large cosine similarity) with the ground truth labels of the corresponding bounding box, while having as low as possible overlap with the ground truth labels of other bounding boxes in the batch (even for predictions within the same image). The contrastive loss is

implemented similarly to [1]. A more in-depth explanation of the contrastive loss used is provided in section 7.1.

The total loss used for backpropagation is the sum of these 2 losses.

Interactive Training

The original SAM implementation [12] uses an interactive training recipe whereby the model is prompted multiple times, with each successive prompt being in the error region of the previous prediction. This was also tried in Semantic-SAM's training, but yielded no gain in mask quality or semantic ability, so it was dropped to keep training simple. This is likely because interactive training is beneficial in improving a model's ability to create segmentation predictions, but the goal here is to only maintain this ability.

3.4.5 Making the model ambiguity aware

Similar to the original SAM implementation [12], Semantic-SAM outputs multiple mask and semantic predictions in each training iteration in an attempt to make the model handle ambiguity in the prompt. This is necessary as a prompt can and should refer to multiple valid masks and corresponding semantic labels. Consider Figure 3.8 for example, a point-query on the vehicle's wheel, can refer to either the wheel alone or the whole car. In this case, multiple masks should be produced, and the corresponding semantic label should align with the produced mask and not the prompt.

Figure 3.8: Ambiguity in a prompt

To achieve this, the original SAM method of making the model ambiguity aware was expanded on; for a combination of prompt(s) and image, the model predicts 3 masks and 3 semantic labels.

During training, the loss for each of the 3 predictions is calculated, and only the smallest loss contributes to the total loss. This mechanism encourages each token to learn to predict a mask and corresponding semantic prediction at a different granularity (the tokens predict a mask and semantic label that corresponds to part, subpart, and whole object).

3.4.6 Training Infrastructure

Training occurred on ETH's Euler Cluster [34], the exact GPUs used depended on what was available at the time, but were always one of: TITAN RTX, Quadro RTX 6000 and RTX3090. Only 24 GiB GPUs were used.

Due to the design of the file system (permanent storage is attached via the network), data had to be aggregated into shards (otherwise this would lead to many reads to small files, which would be extremely slow), each containing $\sim$ 3400 samples for a total of $\sim$ 1100 shards. No shuffling was performed above the shard level, each shard simply contained the image-text pairs from a single LAION URL shard post filtering (each LAION shard contains 10k URLs, around 10% - 15% of these will fail to download, the rest are filtered based on language, NSFW content or because no confident bounding boxes

were detected). SA-1B data was similarly sharded, as this is how SA-1B is distributed by Meta, this simply required sorting each of the SA-1B archive files.

For simple distributed training and to avoid manually copying and extracting shards to a node's local disk, the WebDataset library [38] was used to load data, with data being distributed among the individual GPUs by splitting at the shard level (the basic 'split by node' distribution strategy was applied, where the i-th GPU reads from the shards $i, i+n, i+2n, ...$ where n is the total number of GPUs).

3.4.7 Fine-Tuning

Fine-tuning on instance-segmentation data occurred similarly to the main training phase, and only differed in the fact that only one data source was used within a batch, with this data used to supervise both the mask predictions and the semantic predictions.

3.5 Inference

SAM produces a semantic prediction as a vector S, for downstream use cases, masks typically have to be classified as one of a number of possible classes $\{c_1, c_2, .., c_n\}$. Figure 3.9 outlines the method of inference. Queries (either text or images) are encoded by the respective CLIP encoder, this results in a set of query embeddings $\{C(c_1), C(c_2), .., C(c_n)\}$. Computing the Cosine Similarity between the model's prediction and encoded classes. Subsequently, applying a soft-max, produces per class logits.

To increase the robustness of classification, multiple prompts should be supplied per class. In the case of images, this can be done by encoding multiple images of the same class, for text, this is done by wrapping the class label in multiple text prompts. The prompt templates listed in section 7.6 lists templates for most of the examples shown.

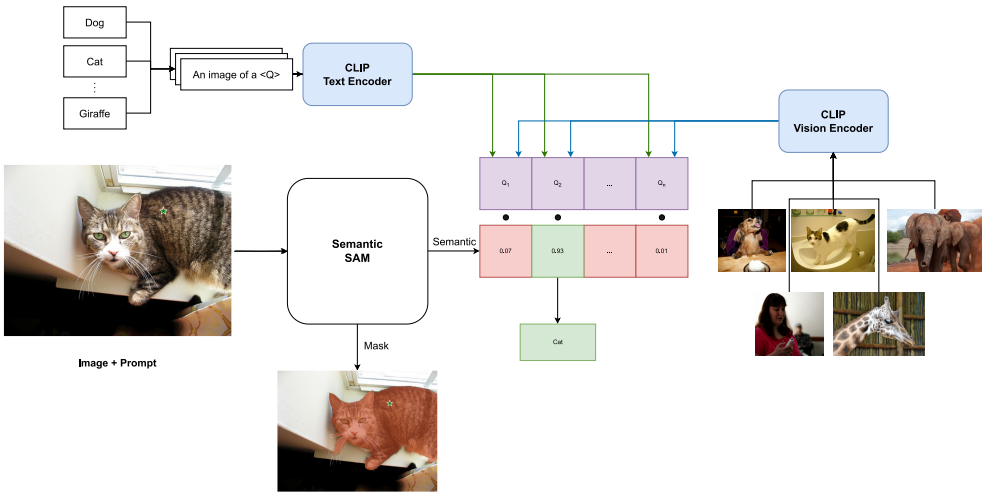


Figure 3.9: Overview of inference method.

Chapter 4

Results

This chapter delves into the results derived from the method highlighted previously in chapter 3. First, A visual showcase of Semantic-SAM's capabilities is presented, especially in scenarios where a quantitative assessment falls short. Subsequently, Semantic-SAM's performance metrics when evaluated on two instance segmentation benchmarks, namely SEGINW and COCO, is presented.

4.1 Qualitative Results

4.1.1 Mask Quality

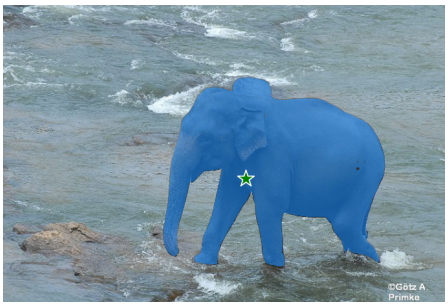
By basing the model on SAM, it was important to maintain SAM's segmentation ability. Figure 4.1 compares SAM's mask predictions (Left) with Semantic-SAM's predictions (Right) for the same image and prompt, showing no visible reduction in mask quality.

(a) SAM

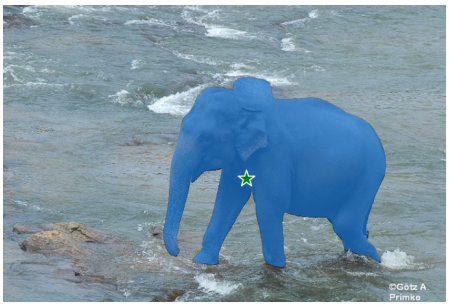
(b) Semantic-SAM

(c) SAM

(d) Semantic-SAM



(e) SAM



(f) Semantic-SAM

Figure 4.1: Semantic-SAM mask outputs compared to SAM (both using the ViT-L backbone)

4.1.2 Prompt Ambiguity

A property which relevant for downstream applications is the ability of Semantic-SAM to produce multiple masks, to resolve ambiguity in a prompt. When multiple masks are generated, the predicted semantic embedding should in this case align with the semantic label of the object the mask covers and not the prompt. For example, given a point prompt on a wheel, the model will produce masks that cover the wheel, and the whole car, the mask associated with the wheel should be accompanied by a semantic prediction aligned with `'wheel'` and not `'car'`.

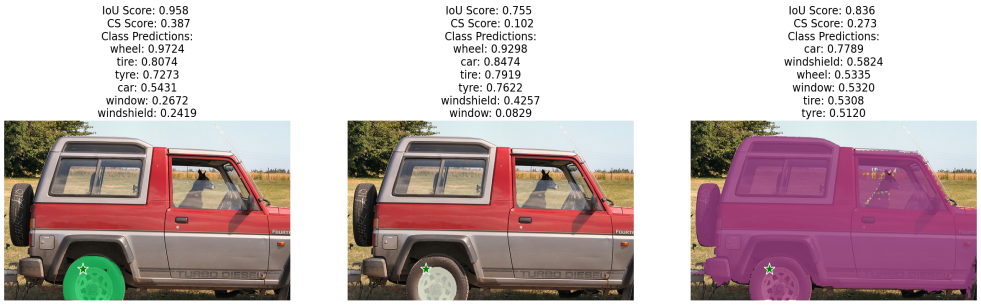


Figure 4.2: Point query on a Car's wheel and associated CS with different queries

This is an area in which Semantic-SAM does fall slightly short. This occasionally manifests itself in semantic predictions of sub-parts that are more aligned with the semantic label of the whole part, as shown in Figure 4.3, where both the Wing, engine and rudder are assigned the label `'aircraft'`.

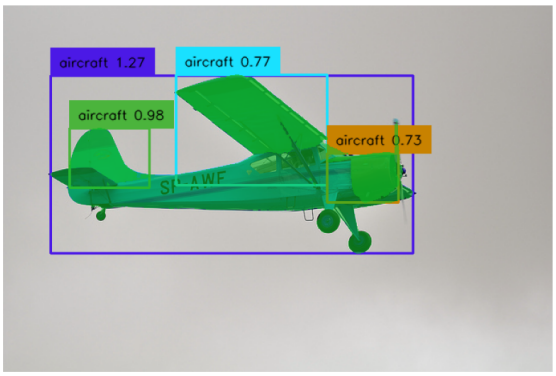


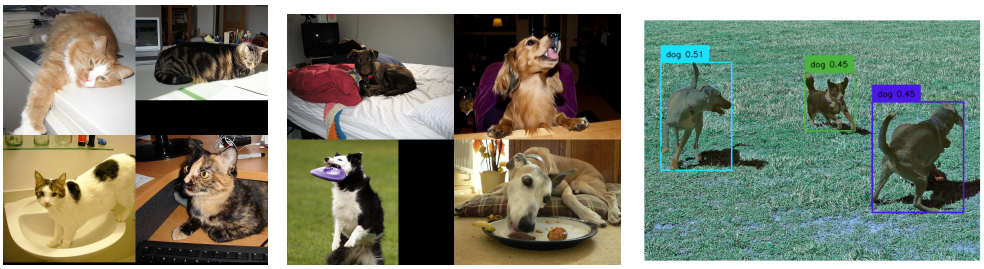
Figure 4.3: Queries: `'airplane'`, `'airplane wing'`, `'airplane rudder'`, `'airplane engine'`

Another example of such a failure is presented in Figure 7.2. The issue impacts downstream use cases, as sub-parts with a wrong labels are not easily filtered based on confidence or CS-score.

4.1.3 Multi Modal Queries

Semantic-SAM's training data was created by encoding an instance's label using the CLIP text-encoder, as CLIP is a multimodal model, this means the predictions made will also be aligned with the corresponding vision encoder's output. This allows classification of masks, either by classifying against a query image or query string.

In practice however, text queries tend to perform better than image queries, though this limitation can be mitigated slightly by supplying multiple query images, similar to the way multiple prompt templates are used for a single text query. This is shown in Figure 4.4, which provides some example query images (Left), (Middle) and the result of querying an image (Right).



(a) `'Cat'` query **(b)** `'Dog'` query **(c)** results on target image

Figure 4.4: Querying an image with image embeddings

4.2 Quantitative Evaluation

In this section, Semantic-SAM's evaluation is presented in terms of its performance on 2 segmentation benchmarks, Segmentation in The Wild (SEGINW) and COCO. This will show the strengths and shortcomings of the model and training data generation method.

4.2.1 Evaluation Method

SEGINW and COCO are instance segmentation benchmarks, for a given image, the model under evaluation provides a sequence of masks and associated category and confidence predictions.

To generate these predictions across the whole image, evaluation is performed similarly to [12], Semantic-SAM is prompted with a grid of points as shown in Figure 4.5, each point produces 3 masks. If not otherwise specified, a 16×16 grid is used, this results in up to 768 masks.

Each mask is then classified, to classify a mask, the Cosine Similarity between the predicted CLIP embedding and CLIP embeddings of all the possible classes is computed, to make the classifier more robust, for each class, multiple classifier embeddings are computed, each with a different prompt, similar to [2], the prompt templates used to produce classifier embeddings are provided in section 7.6, in total, there are 5 templates, increasing the number of prompts beyond 5 did not provide any benefit. If the cosine similarity between the predicted embedding and any of the prompt embeddings is above a certain threshold (0.6 used in most cases) a mask is considered to belong to that category, other masks are discarded at this point.

To remove duplicates, non-naximum suppression is applied, and low confidence masks are filtered out by filtering on predicted IoU, predicted CS and stability score. The stability score calculation from the original SAM implementation is used. The exact parameters and filtering thresholds used are provided in Appendix 7.5. This results in a set of high confidence masks, which are output as the model's segmentation predictions.

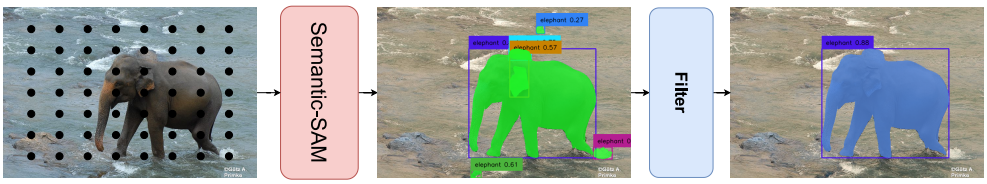


Figure 4.5: Evaluation

4.2.2 Segmentation In the Wild (SEGINW)

As the Segmentation In the Wild benchmark was designed to measure a model's ability to perform on various downstream tasks with little or no training data, it serves as a good benchmark to inspect Semantic-SAM's abilities. The benchmark is described in detail in section 2.4.4

Results

Three evaluation results are provided: 2 entries for the Zero-Shot category and 1 entry for the Full-Shot Benchmark. For the first Zero-Shot benchmark, Semantic-SAM is trained only on the 3.9M image dataset described in chapter 3, for the Full-Shot entry, Semantic-SAM is further fine-tuned on the training split of SEGINW. For the second Zero-Shot entry, Semantic-SAM is fine-tuned on a small dataset, generated identically to the 3.9M images, but chosen in such a way that it is skewed towards including the SEGINW categories.

Zero Shot: Without fine-tuning, as is shown in Table 4.1, Semantic-SAM achieves strong performance on a variety of the sub-benchmarks, even though the model was not explicitly trained on any of the classes present in these datasets. But the aggregated performance over all datasets is poor, achieving **19.56** mAP, lower than all the baseline implementations provided by [19] of similar model size.

Full-Shot: This of course raises the question, what is the cause of this comparatively low performance? The model architecture, training method and dataset are all new or at least heavily modified, so to determine which is the cause of this low performance, Semantic-SAM was first fine-tuned on the SEGINW training split to rule out model architecture or training objective as a culprit.

When fine-tuned on SEGINW, Semantic-SAM achieves **46.69** mAP on average on SEGINW, this outperforms all Full-Shot entries published as of writing this thesis, and is outperformed only by a combination of Grounding-DINO and SAM ViT-H (Bounding boxes predicted by Grounding-DINO, which are used to prompt SAM). The results of Semantic-SAM on each dataset are provided in Table 4.1.

Dedicated Data The strong Full-Shot results clear the model architecture and training method as potential culprits. This leaves the training data as the likely cause of Semantic-SAM's low zero-shot performance.

The training data can cause this in 2 ways; either the data contains too few training samples for the classes present in SEGINW or there is an underlying issue in the way the data is generated. The results on SEGINW suggest the former, as Semantic-SAM performs reasonably well on classes that can be expected to be common in a dataset scraped from the internet, such as `'Strawberry'` and `'Elephant'`.

This assumption is further strengthened by looking at the number of appearances these make in the training dataset reported in Table 4.1. All text embeddings of the 3.9M dataset were compared to the embeddings of the SEGINW classes. For each dataset, 4 numbers n_{th} are reported, where th is the respective threshold, meaning all instances with Cosine Similarity above this threshold were counted. [1]

As there are over 85 classes in SEGINW, the comprehensive table where the frequency of each category is listed separately is provided in Table 7.2, while Table 4.1 reports only the average number of occurrences per dataset.

There is at least a loose indication that too few training samples could be the cause of this low performance.

We now ascertain whether this is caused by the data generation pipeline (This could happen, for example, due to Grounding-Dino's inability to find certain classes, or bias causing the Language Model to not propose certain objects) or because there just are not enough training samples for each class at the start of training.

This can be examined by gathering a dataset that is heavily skewed towards the SEGINW classes. For each SEGINW category 10k images were sourced from LAION by querying the dataset using the clip-retrieval tool [6], these images and captions were then used to generate a dedicated dataset. Importantly, the annotation pipeline was not altered in any way as to bias it to make object proposals, image-text pairs are treated identically to how normal LAION image-text pairs are. This results in a dataset of 532k images, per category occurrences are listed in Table 7.4, Although queries with training images, this failed to adequately retrieve some classes.

[1] For context, using the same text encoder as was used in the data generation, the words `'Dog'` and `'Puppy'` have a cosine similarity of 0.86 while `'fridge'` and `'refrigerator'` have a similarity of 0.97.

After fine-tuning on this small dedicated data, Semantic-SAM achieves **22.6 mAP** on SEGINW. As the images are sourced from LAION and do not overlap with the training or validation sets, this can still be considered to be zero-shot performance according to the benchmark rules.

This performance increase, compared to the zero-shot results, provides evidence that the low performance is (partially) attributable to the low coverage of SEGINW classes in the 3.9M image dataset and points away from an underlying issue with the data generation method.

	X-Decoder Zero Shot	X-Decoder Full Shot	S-SAM Zero Shot	S-SAM Dedicated Data	S-SAM Full Shot	$f_{0.99}$	$f_{0.95}$	$f_{0.9}$	$f_{0.8}$
Average	32.2	44.6	19.6	22.7	46.7	-	-	-	-
Elephants	66.0	72.8	68.0	60.0	51.8	5677.0	5916.0	7352.0	13497.0
Hand-Metal	42.1	67.8	54.1	46.7	65.8	31076.0	31914.0	41643.5	64950.5
Watermelon	13.8	44.3	25.5	34.5	53.1	1276.0	1277.0	1497.0	29344.0
House-Parts	7.0	11.8	0.8	1.2	7.5	29.3	42.6	1446.9	20129.6
HouseHold-Items	53.0	52.7	25.0	50.0	17.7	25877.0	26028.8	33325.2	71786.8
Strawberry	67.1	90.7	75.9	75.3	89.9	12639.0	12639.0	22955.0	150976.5
Fruits	79.2	78.9	4.6	15.6	24.6	8697.2	9247.2	10485.2	23951.0
Nutterfly-Squireel	68.4	84.3	22.3	10.2	52.0	13616.0	13815.0	15800.0	21943.0
Hand	75.9	90.2	0.0	61.5	99.0	28883.0	29660.0	39243.0	58849.0
Garbage	33.0	42.0	9.2	3.9	34.9	2851.5	3005.2	3221.0	7033.8
Chicken	8.6	8.8	53.0	56.5	93.1	4856.0	5136.0	5513.0	10394.0
Rail	2.3	56.8	0.2	1.3	67.8	767.0	805.0	1018.0	2911.0
Airplane-Parts	13.1	13.6	5.0	3.0	7.2	2742.6	3709.0	4310.8	8171.6
Brain-Tumor	2.2	4.2	0.0	4.3	29.3	82.0	82.0	94.0	124.0
Poles	20.1	21.6	10.0	10.8	20.4	140.0	171.0	216.0	1594.0
Electric-Shaver	7.5	51.1	15.3	15.8	74.6	0.0	0.0	0.0	9.0
Bottles	42.1	49.3	7.6	4.9	50.5	29607.7	29802.3	37621.0	56945.3
Toolkits	9.9	11.1	5.0	1.1	59.9	630.1	671.1	917.2	5318.4
Trash	22.3	39.7	3.6	1.3	13.1	106.2	150.6	354.2	18692.2
Salmon-Fillet	19.0	50.4	1.8	5.1	24.6	0.0	12.0	44.0	1791.0
Puppies	59.0	54.3	76.0	73.2	76.5	408.0	736.0	1189.0	46169.0
Tablets	22.5	42.1	0.0	17.2	43.2	220.0	220.0	2626.0	7206.0
Phones	15.6	18.9	8.1	0.0	26.3	8768.0	9124.0	14668.0	116742.0
Cows	44.9	44.6	1.9	1.3	47.0	939.0	1240.0	1533.0	38581.0
Ginger-Garlic	11.6	13.8	16.2	11.8	38.4	697.5	741.5	818.0	1816.0

Table 4.1: Performance on SEGINW showing mAP (left) of different entries and average number of samples per SEGINW dataset

4.2.3 COCO

Semantic-SAM's instance segmentation ability was also evaluated on COCO. When trained only on the 3.9M image dataset described in chapter 3, Semantic-SAM's performance on COCO is extremely poor, achieving only **1.2** mAP.

As with SEGINW, the link between the number of training samples and Semantic-SAM's performance was analyzed. Table 7.1 reports the number of samples per class vs. the Average Precision for the 80 coco classes.

	COCO mAP A	COCO Recall @ 100
3.9M Image Dataset	1.2	4.1
Dedicated Dataset	5.1	16.3

Table 4.2: Semantic SAM's performance on COCO

Dedicated Data

Similarly to section 4.2.2 a dedicated dataset is constructed for COCO. The approach used in section 4.2.2 with the individual images being used to query CLIP-retrieval, however, is less effective in this setting as COCO emphasizes contextual diversity and leads to mainly unrelated images. Instead, the class labels were used. All class labels are wrapped in the CIFAR-10 prompts found in [39] in an attempt to retrieve a more varied assortment of images. For each class label, CLIP-retrieval provided $\sim$ 15k URLs, or $\sim$ 1.2M URLs in total. After downloading, these images are again treated the same as the original LAION data (no prior information about the possible class affiliations is provided during pseudo label generation). After download failures and filtering based on image metadata and low confidence pseudo labels, this results in a dataset of $\sim$ 680k pseudo labeled images.

Table 7.3 shows the distribution of this new dataset, this method of collecting data resulted in a more uniform distribution across classes than the approach taken in SEGINW Table 7.4. As in the SEGINW benchmark, training on this comparatively smaller dedicated dataset leads to a significant increase in the model's mAP and recall on COCO, increasing mean Average Precision to **5.1** mAP. This result is still low and the likely cause of this is outlined in section 4.2.4

4.2.4 Discussion of Results

Performance gap

There is a significant gap in performance between Semantic-SAM's performance on SEGINW and COCO, both in the zero-shot setting and when providing skewed data.

A likely cause of this is due to the difference in the 2 benchmarks. SEGINW emphasizes diversity in the class labels, featuring categories not present in major detection datasets, which lets it assess a model's ability to generalize to unseen data, while COCO focuses on contextual diversity and the ability to detect potentially obscured objects in their natural context.

The method in which training data is collected from LAION, results in a dataset which is dominated by iconic images, as this is what is common on the internet. For example, the query `'Bottle'` (Here) on LAION produces almost entirely images of single bottles on a neutral background, in COCO however, bottles are pictured in their natural context (fridges, tables, bars, etc.) and are often only partially visible, examples of this are shown in Figure 4.6.

Figure 4.6: Example COCO images with bottles

In comparison, although SEGINW features a diverse set of categories, the images rarely exhibit the contextual diversity seen in COCO, 3 examples are shown in Figure 4.7

Figure 4.7: Example SEGINW images with bottles

Effect of skewing Data content

As shown above, Semantic-SAM trained on a domain-specific dataset significantly boosts the performance in a zero-shot setting. This indicates that much larger subsets of LAION would be needed to build a truly universally capable zero-shot model.

Small Objects

Further, as shown in Table 4.3, Semantic-SAM's performance is much higher on large and medium-sized objects compared to small objects for both COCO and SEGINW. The cause of this is, in all likelihood, the evaluation method, a grid of points is more likely to pick up a large item than it is a small item. Increasing the number of points did not increase the accuracy of the model.

	Small	Medium	Large
SEGINW Zero Shot	2.37	10.79	31.0
SEGINW Dedicated Data	2.1	12.5	30.1
SEGINW Full Shot	5.16	34.9	52.1
COCO Zero Shot	0.4	1.5	3.1
COCO Dedicated Data	0.5	4.0	12.2

Table 4.3: mAP by ground truth size

Chapter 5

Limitations & Future Work

In this chapter, we provide a comprehensive overview of some of the notable limitations that have been discussed in the preceding chapters. Moreover, we explore potential strategies and methods that could be employed to address and potentially overcome these limitations in subsequent research and work.

5.1 Spatial understanding

An ability missing in Semantic-SAM is the ability to understand object positions. The data generation pipeline was not specifically designed to allow for spatial understanding propagating into the data labels. Example queries in Figure 5.1 show that queries match the overall object, but fail to incorporate the spatial descriptions in the text.

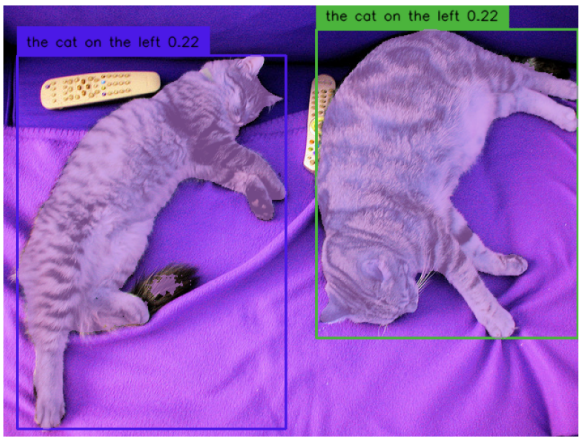


Figure 5.1: Queries: ['the cat on the left', 'the cat on the right']

Another example of such a failure is presented in Figure 7.3

5.2 Scaling Dataset Size

As shown in chapter 6, Semantic-SAM's zero shot capabilities are not impressive when only trained on the 3.9M images generated from the LAION subset. On both COCO and SEGINW, fine-tuning on a comparatively smaller dataset with pseudo annotations generated in the exact same way, but with image-text pairs sourced from a subset of LAION skewed to include the relevant evaluation categories, significantly improves Semantic-SAM's performance.

To build a dataset that would allow Semantic-SAM to exhibit strong zero-shot performance across a wide variety of scenarios, requires scaling the size of the dataset, this has shown to work for approaches such as OWL-ST [3], where the authors report increasing performance up to billions of images.

5.3 Scaling Model Size

The SAM version used for all experiments provided in chapter 6 is the ViT-L Backbone model. SAM was released in 3 versions, (ViT-B, ViT-L, ViT-H), the ViT-L model was chosen primarily for ease of training[1] and because the authors of SAM report that the gain in segmentation ability when going from ViT-L to ViT-H was significantly smaller than the gain from going from ViT-B to ViT-L.

[1]The ViT-H is too large to be trained on a 24GPU card, making training impractical on Euler[34]

5.4 Image Source

Orthogonal to scaling dataset and model size is changing the source of image-text pairs. As mentioned in section 4.2.4, the current source of images is dominated by iconic images featuring single objects alone with a neutral background, with little diversity in object context.

LAION, which was used to derive all training data, [31] is curated by scraping the `HTML` content of websites for image-text pairs, these websites are listed on Common Crawl. The ViT-B/32 CLIP model is then used to compute image and text embeddings, image-text pairs for which the cosine-similarity between these 2 embeddings is below 0.28 (0.26 for non-English text) are dropped at this point. Importantly, there is no mechanism to ensure contextual richness of images.

As such, LAION, and by extension the dataset used for Semantic-SAM, is dominated by iconic images, reflecting the types of images frequent on the internet. As I do not own the copyright to any of these images, examples can be viewed Here.

This issue was already identified and addressed in COCO, where the curators explicitly collect only non-iconic images, and focus on images in which the target objects appear in their natural context. In COCO, they do this by querying Flickr with combinations of class labels, such as querying “dog” + “bottle” instead of just querying “dog”. It is possible to apply a similar mechanism when collecting images from LAION, but this requires prior knowledge of the categories and cannot be naively extended to an open-world setting.

Collecting vast image-text pairs to derive training samples with COCO style high contextual diversity in an open-world setting would likely require careful filtering of these image-text pairs prior to annotating them. How exactly to approach this problem is not immediately obvious, a possible approach could be to analyze the variability of colors in the border of an image and disregard images with low variability. This will filter images in which there is a single large object with a neutral background.

Again, such a mechanism may not even be necessary when scaling the dataset to much larger scale, as Vision-Language models have shown the ability to learn from such data.

5.5 Language Model

The language model and method of proposal extraction used arguably has the greatest impact on every aspect of the generated dataset. This is because it impacts every subsequent stage of data generation, any limitation or bias here is compounded in later stages. An aspect of this thesis that was not further explored was the effect of using different language models to generate object proposals during data generation. The 7B parameter [32] BigScience model was chosen mainly out of convenience, as it supported efficient inference with vllm [35] and it fit on a single 24Gb GPU, making inference for millions of queries feasible.

Switching to a more performant and larger model results in higher quality proposals and possibly also less biased object proposals.

Orthogonal to switching to a stronger LLM, methods to enhance proposals, such as fine-tuning or few-shot prompting, may also be worth exploring. The LLM used in the data generation was not fine-tuned. Experiments with fine-tuning using both plain COCO and RefCOCO labels resulted in proposals heavily skewed towards COCO classes.

Similarly, few shot prompting can aid in guiding the LLM to make better predictions, for example: [2] [3]

```
<human>: An image has the description 'A bunch of fruits hand on a branch'
<bot>: objects likely visible in the image are: fruit, pear, apple, orange, peach, tree, leaf,
farmer, field
<human>: An image has the description 'a woman running up to a tennis ball to hit it',
<bot>: objects likely visible in the image are: tennis player, tennis racket, tennis ball,
<human>: An image has the description 'Kay Coughlan and Rosaline Higgins pictured at Bressies
Charity lunch in aid of Larcc Cancer Support in Brasserie 15 Restaurant in Castleknock',
<bot>:  objects likely visible in the image are:  people, man, woman, hat, jacket, shirt, tie,
dress, chair, table, restaurant, room, building, window, ceiling, wall, floor, light, lamp,
chandelier, plant, food, drink, glass, cup, saucer, plate, bowl, spoon, fork, knife, napkin, menu,
book, newspaper, magazine, mirror, painting, picture frame, clock, door, sign, poster, advertisement,
computer, laptop, monitor, keyboard, mouse, phone, camera, flash, bottle, can, bag, purse
```

In general, few-shot prompts produce more relevant proposals, but the choice of few-shot examples will significantly influence the types of proposals generated. How to leverage few-shot prompting without introducing bias into the types of proposals made must be studied in detail.

[2] Text taken from LAION, used as an example in section 3.1
[3] Produced by LLama-70B

5.6 Bias

In the scope of this thesis, for both the language model and the open-vocabulary object detection model, bias was not taken into account, these models however will exhibit some bias.

For both language models and vision models, much research has shown the current generation of Large Language Models [40, 41] and vision models [42, 43] exhibit bias reflecting the data they are trained on. Though this research typically explores the stereotypical biases concerning race, gender, ethnic and religious attributes, this bias will also extend in a way that impacts the quality of pseudo labels generated.

5.6.1 Language Model Bias

The bias of a language model will affect what kinds of candidates are produced. Consider for example the caption `'Black Students at sports practice'`, in the data generation pipeline described in chapter 3, 3 of the 5 prompt templates produce the proposal `'Basketball player'`. Switching ethnicities to `'Caucasian students at sports practice'`, not a single proposal for `'Basketball player'` is returned.

This bias also presents itself in terms of gender, for the caption `'engineer working on a computer'`, 2/5 prompts produce the proposal `'man'`, none produce `'woman'` or any feminine descriptions. The caption `'A scientist'` produces the proposals `'man'` and `'male scientist'` while not returning any feminine proposals. Similarly, `'A professor in a lecture hall'` also returns `'man'` while not returning any female proposals.

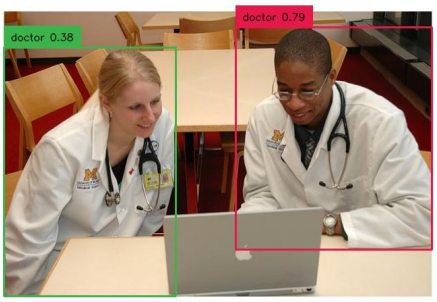
This bias extends beyond human identity. For example, providing the caption `'Eating utensils'` or variations of it, predominantly produces proposals of eating utensils common in Europe and North-America, such as: `'fork'`, `'spoon'`, `'knife'`. The prompt had to be engineered to `'Eating utensils used around the world'` to get 1/5 prompts to return `'Chopsticks'`.

A possible workaround could be to use prompts in different languages.

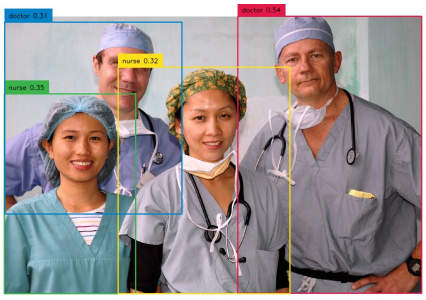
5.6.2 Detection Model Bias

The open-vocabulary object detector will further introduce bias in the form of what objects are detected, and what label they are assigned. Consider Figure 5.2, Given the query `'Doctor'` and `'Nurse'`, Grounding-DINO systematically assigns the label `Doctor` to male individuals, while labeling the female individuals with the label `Nurse`. When limiting the queries to just `'Doctor'`, Grounding-DINO assigns higher confidence to the male individuals. Similarly, when presented an image of Marie and Pierre Curie (Both Nobel Prize winning Scientists), Grounding-DINO assigns Marie Curie the label `'assistant'`, when given the choice: `'Scientist', 'Assistant'`.

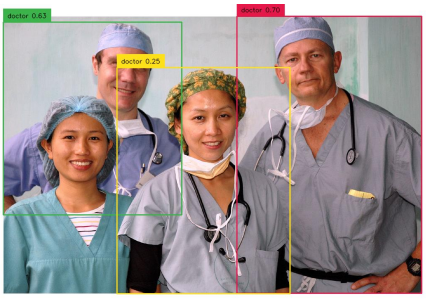
(a) Query: `'Doctor', 'Nurse'`



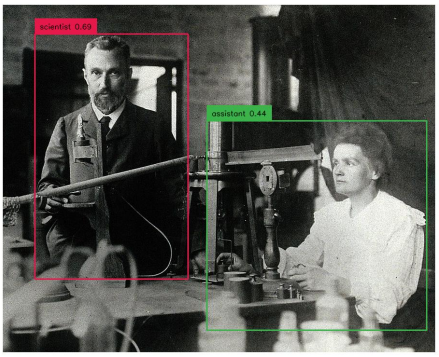
(b) Query: `'Doctor'`



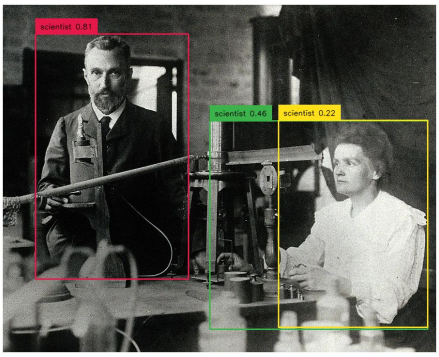
(c) Query: `'Doctor', 'Nurse'`



(d) Query: `'Doctor'`



(e) Query: `'Scientist', 'Assistant'`



(f) Query: `'Scientist'`

Figure 5.2: Examples of bias in pseudo annotations

5.7 Image to Text Models

Recent advances in the realm of Image-to-Text modelling has produced models capable of generating accurate captions given an image [44, 45]. Adding such captions to give the language model more information to work with could increase the relevance and quality of object proposals, however, such an addition to the annotation pipeline will undoubtedly introduce further bias into the dataset.

5.8 Self Training

An approach that has shown to work well in [3, 28, 12] is self training, wherein a model is first trained on existing datasets, then it is used to label more data that can be used to retrain the base model. In the Context of Semantic-SAM, this would mean using Semantic-SAM to generate labels for any further data. This will very likely result in a stronger dataset, as chapter 6 demonstrated Semantic-SAM's ability to outperform grounding-dino on categories not well represented in grounding-dino's training data.

Chapter 6

Conclusion

With an introduction into the current approaches taken to tackle the task of object detection in chapter 2, I provided a brief overview of the methods found to work well.

Following recent trends in Machine Learning, much focus has been applied to the goal of expanding the size and variability of object detection and instance segmentation datasets. In section 2.5, I introduce the reader to the previous approaches taken in this goal. The most successful of these methods take a weakly supervised approach and derive pseudo labels from an image's caption.

This sets the stage for section 3.1, which delves into a limitation present in many of these approaches. This limitation arises from a natural rift between the type of information humans typically provide in an image caption and the information needed to produce relevant and unbiased candidates for pseudo labels.

In chapter 3, I introduce an alternative approach of deriving these pseudo labels. By going beyond only the semantic content of an image's caption and by leveraging language modelling to predict further adequate proposals, we create a method of generating weakly supervised datasets that include a wider variety of labels.

Subsequently, chapter 4 presents the effectiveness of this approach, both with visual examples and quantitative evaluation on 2 common instance segmentation benchmarks. The benchmarks show that, although not designed specifically for the task, Semantic-SAM is able to achieve competitive results in a Full-Shot setting, achieving **46.7 mAP** on SEGINW. The Zero-Shot experiments as well as the benchmark on COCO, however, reveals a weakness in the method of curating the dataset. This weakness and further limitations are presented in chapter 5, which outlines the limitations of our approach and possible methods of mitigating these in the future.

Bibliography

[1] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[2] M. Cherti *et al.*, "Reproducible scaling laws for contrastive language-image learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2818–2829.

[3] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," *arXiv preprint arXiv:2306.09683*, 2023.

[4] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[5] A. Ramesh *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[6] R. Beaumont, *Clip retrieval: Easily compute clip embeddings and build a clip retrieval system with them*, https://github.com/rom1504/clip-retrieval, 2022.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[11] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[12] A. Kirillov *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[14] J. Wu *et al.*, "Medical sam adapter: Adapting segment anything model for medical image segmentation," *arXiv preprint arXiv:2304.12620*, 2023.

[15] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[16] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, "Modeling context in referring expressions," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, Springer, 2016, pp. 69–85.

[17] D. Zhao, A. Wang, and O. Russakovsky, "Understanding and evaluating racial biases in image captioning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 830–14 840.

[18] A. Kuznetsova *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[19] X. Zou *et al.*, "Generalized decoding for pixel, image, and language," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 116–15 127.

[20] *Papers with Code - ImageNet Benchmark (Image Classification)*, [Online; accessed 10. Sep. 2023], Sep. 2023. [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet.

[21] A. Fang *et al.*, "Data determines distributional robustness in contrastive language image pre-training (clip)," in *International Conference on Machine Learning*, PMLR, 2022, pp. 6216–6234.

[22] V. Smil, "How Many People Did it Take to Build the Great Pyramid?" *IEEE Spectr.*, Jul. 2021. [Online]. Available: https://spectrum.ieee.org/how-many-people-did-it-take-to-build-the-great-pyramid.

[23] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," in *European Conference on Computer Vision*, Springer, 2022, pp. 350–368.

[24] A. Gupta, P. Dollar, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5356–5364.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[26] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2556–2565.

[27] Y. Zhong *et al.*, "Regionclip: Region-based language-image pretraining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 793–16 803.

[28] R. Arandjelović, A. Andonian, A. Mensch, O. J. Hénaff, J.-B. Alayrac, and A. Zisserman, "Three ways to improve feature alignment for open vocabulary detection," *arXiv preprint arXiv:2303.13518*, 2023.

[29] X. Chen *et al.*, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[30] *laion/laion400m · Datasets at Hugging Face*, [Online; accessed 13. Sep. 2023], Sep. 2023. [Online]. Available: https://huggingface.co/datasets/laion/laion400m.

[31] C. Schuhmann *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.

[32] N. Muennighoff *et al.*, "Crosslingual generalization through multitask finetuning," *arXiv preprint arXiv:2211.01786*, 2022.

[33] *bigscience/bloom-7b1 · Hugging Face*, [Online; accessed 29. Aug. 2023], Aug. 2023. [Online]. Available: https://huggingface.co/bigscience/bloom-7b1.

[34] *Euler - ScientificComputing*, [Online; accessed 10. Sep. 2023], Sep. 2023. [Online]. Available: https://scicomp.ethz.ch/wiki/Euler.

[35] *Welcome to vLLM! — vLLM*, [Online; accessed 29. Aug. 2023], Jul. 2023. [Online]. Available: https://vllm.readthedocs.io/en/latest.

[36] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[37] *EvalAI: Evaluating state of the art in AI*, [Online; accessed 30. Aug. 2023], Aug. 2023. [Online]. Available: https://eval.ai/web/challenges/challenge-page/1931/leaderboard/4567.

[38] A. Aizman, G. Maltby, and T. Breuel, "High performance i/o for large scale deep learning," in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 5965–5967.

[39] *CLIP_benchmark/clip_benchmark/datasets/en_zeroshot_classification_templates.json at main · LAION-AI/CLIP_benchmark*, [Online; accessed 12. Sep. 2023], Sep. 2023. [Online]. Available: https://github.com/LAION-AI/CLIP_benchmark/blob/main/clip_benchmark/datasets/en_zeroshot_classification_templates.json.

[40] A. Abid, M. Farooqi, and J. Zou, "Persistent anti-muslim bias in large language models," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 298–306.

[41] M. Nadeem, A. Bethke, and S. Reddy, "Stereoset: Measuring stereotypical bias in pretrained language models," *arXiv preprint arXiv:2004.09456*, 2020.

[42] J. Wang, Y. Liu, and X. E. Wang, "Are gender-neutral queries really gender-neutral? mitigating gender bias in image search," *arXiv preprint arXiv:2109.05433*, 2021.

[43] N. Dehouche, "Implicit stereotypes in pre-trained classifiers," *IEEE Access*, vol. 9, pp. 167 936–167 947, 2021.

[44] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.

[45] A. Awadalla *et al.*, "Openflamingo: An open-source framework for training large autoregressive vision-language models," *arXiv preprint arXiv:2308.01390*, 2023.

Chapter 7

Appendix

7.1 Contrastive Loss

The contrastive loss is virtually identical to the Open-CLIP [2] implementation. The n predicted embeddings s_i are normalized and then multiplied with the n ground truth embeddings T_i to produce a cost-matrix M, the entry m_{ij} is the cosine similarity between predicted embedding i and ground truth embedding$j$.

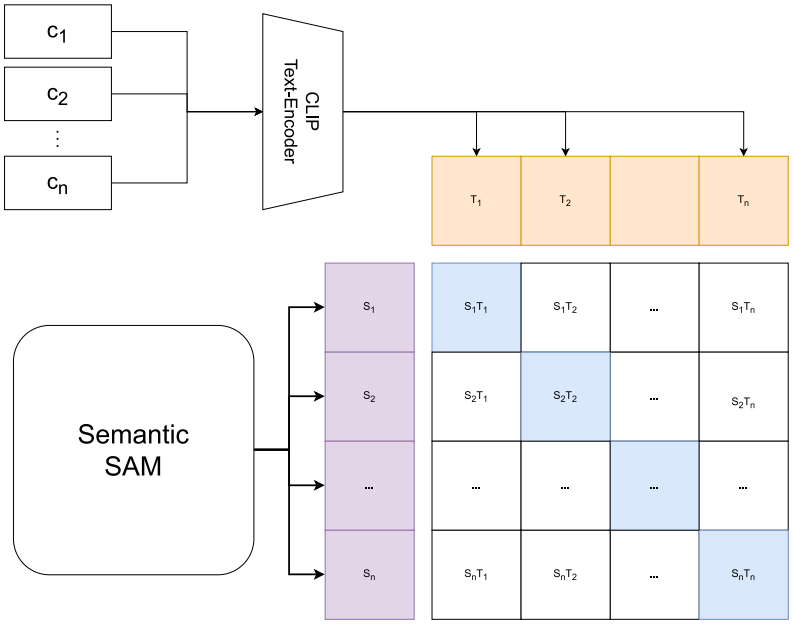


Figure 7.1: Contrastive Loss Cost Matrix

As, the goal of the contrastive loss is to produce embeddings with large overlap with the ground truth, while having a low overlap with the other embeddings in the batch. This is achieved by maximizing the values along the diagonal while minimizing the absolute value of non-diagonal entries. This can be quantified by calculating the cross entropy between the row i and column i and the basis-vector with a one in the $i-th$ element and zero everywhere else. This is done for each row and column, and the average is the contrastive loss:

$$L_{contrastive} = \frac{1}{2 \cdot n}(\sum_{i=o}^{n} L_{CE}(M_i, e_i) + \sum_{i=o}^{n} L_{CE}(M_i^T, e_i)) \tag{7.1}$$

7.2 Examples & Failure Cases

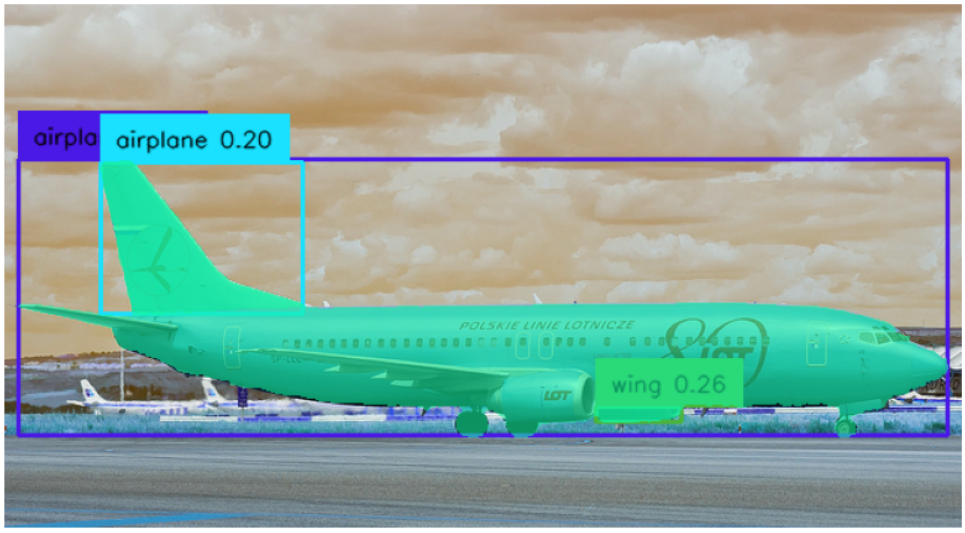


Figure 7.2: Common failure mode: sub-parts are assigned labels of the whole-part
Queries are: ["Airplane", "wing", "rudder", "engine"]

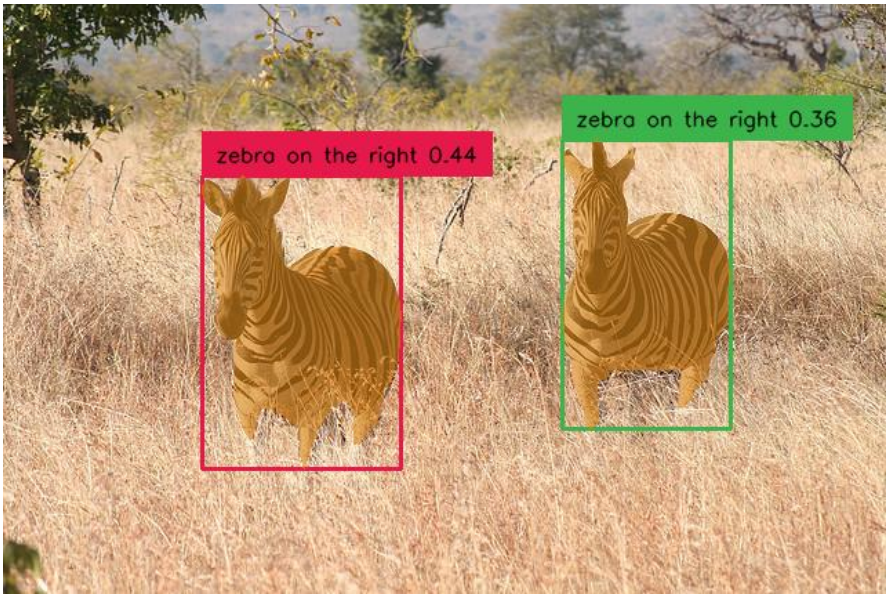


Figure 7.3: Common failure mode: missing ability to understand position
Queries are: ["Zebra on the right", "Zebra on the left"]

7.3 Grounding DINO thresholds

- box_threshold: 0.23
- text_threshold: 0.23[1]

7.4 Training Parameters

The parameters used for training the Semantic-SAM model with a ViT-L Backbone are:

- Batch Size: 5 (4 Semantic Samples + 1 SA-1B sample) [2]
- Learning Rate: 10^{-3}
- Optimizer: AdamW
- Weight Decay: 0.01

7.5 Automatic Mask Generation Parameters

The parameters used for evaluation are:

- points_per_side: 16
- crop_n_layers: 0
- semantic_threshold: 0.7
- pred_iou_thresh: 0.8
- nms_threshold: 0.4
- stability_score_thresh: 0.9
- cosine_sim_pred_threshold: 0.3
- area_threshold: 0.7

7.6 Classifier Prompt Embeddings

The Prompt templates used to produce the classifier are:

- `<Category>`
- `a <Category>`
- `an image of a <Category>`
- `several <Category>`
- `A toy <Category>`

[1] Chosen as they performed well empirically
[2] Batch size limited by available GPUs

7.7 Sample Frequency of COCO Classes

	zero shot mAP	zero shot recall	dedicated data mAP	dedicated data recall	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
person	0.09	0.17	0.14	0.30	245135	245136	255663	311621
bicycle	0.02	0.03	0.03	0.05	9556	13541	22457	57068
car	0.08	0.19	0.08	0.19	299251	299251	311899	428419
motorcycle	0.04	0.11	0.04	0.10	15030	16646	22108	54198
airplane	0.10	0.13	0.15	0.36	5158	9746	11529	21470
bus	0.00	0.00	0.15	0.48	2337	2509	2890	6833
train	0.00	0.00	0.16	0.37	4979	4981	6035	11428
truck	0.01	0.04	0.01	0.06	8838	8839	10075	336073
boat	0.05	0.13	0.07	0.16	25698	25722	27943	47646
traffic light	0.00	0.00	0.03	0.11	220	247	321	1443
fire hydrant	0.03	0.03	0.15	0.32	11	27	36	190
stop sign	0.01	0.07	0.23	0.52	117	136	277	2539
parking meter	0.02	0.01	0.13	0.39	3	8	15	177
bench	0.00	0.01	0.03	0.08	2555	2555	4298	7723
bird	0.01	0.04	0.05	0.15	19283	19283	20665	33659
cat	0.00	0.01	0.15	0.46	18661	18662	22064	27686
dog	0.04	0.26	0.10	0.43	40427	40428	46750	56420
horse	0.02	0.07	0.06	0.19	15039	15041	19455	25922
sheep	0.00	0.00	0.07	0.36	3323	3538	4025	8823
cow	0.01	0.00	0.05	0.24	939	1240	1533	38581
elephant	0.01	0.01	0.13	0.29	5677	5916	7352	13497
bear	0.02	0.08	0.16	0.43	2646	2647	5484	13645
zebra	0.00	0.00	0.17	0.39	860	929	1130	4716
giraffe	0.01	0.04	0.23	0.43	857	957	1268	5638
backpack	0.00	0.01	0.00	0.00	11379	11995	16869	108129
umbrella	0.01	0.05	0.07	0.13	4743	4743	6999	11523
handbag	0.01	0.01	0.00	0.01	11640	13161	90039	198189
tie	0.00	0.01	0.00	0.06	4799	5117	6700	17760
suitcase	0.00	0.00	0.02	0.04	1698	1993	6724	54943
frisbee	0.00	0.01	0.08	0.24	40	46	70	1135
skis	0.00	0.00	0.00	0.00	123	241	2212	5144
snowboard	0.00	0.00	0.00	0.11	457	474	1034	10724
sports ball	0.02	0.05	0.01	0.02	10	19	255	84389
kite	0.00	0.00	0.02	0.15	695	809	983	1214
baseball bat	0.01	0.02	0.00	0.03	128	158	250	18849
baseball glove	0.00	0.01	0.00	0.01	135	153	223	25288
skateboard	0.00	0.00	0.01	0.11	1027	1283	2453	9679
surfboard	0.00	0.00	0.04	0.24	1266	1566	1968	6988

Continued on next page

	zero shot mAP	zero shot recall	dedicated data mAP	dedicated data recall	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
tennis racket	0.02	0.04	0.09	0.15	234	338	515	8934
bottle	0.05	0.15	0.04	0.12	42970	42970	54348	74834
wine glass	0.01	0.12	0.05	0.13	2106	2513	4012	36339
cup	0.01	0.04	0.03	0.12	16463	16464	22016	53232
fork	0.00	0.01	0.01	0.02	1066	1271	1890	5185
knife	0.00	0.00	0.01	0.04	2525	2526	4003	9181
spoon	0.00	0.00	0.01	0.03	2098	2099	3045	5648
bowl	0.02	0.06	0.02	0.08	11247	11248	14873	23696
banana	0.00	0.00	0.00	0.01	2980	3128	4811	17698
apple	0.00	0.05	0.01	0.17	12637	13569	15136	43329
sandwich	0.00	0.00	0.01	0.08	1611	1732	2500	6650
orange	0.00	0.00	0.07	0.19	7574	7945	8107	10397
broccoli	0.00	0.02	0.03	0.18	1419	1662	1934	4351
carrot	0.00	0.00	0.00	0.11	2357	2358	3234	4054
hot dog	0.00	0.00	0.00	0.08	398	573	828	2047
pizza	0.03	0.15	0.14	0.39	3317	3320	3879	7305
donut	0.00	0.01	0.03	0.13	1444	1629	2509	4040
cake	0.02	0.17	0.02	0.08	22728	22740	37473	64123
chair	0.03	0.08	0.03	0.10	124125	124126	145107	232795
couch	0.01	0.05	0.03	0.13	29964	37855	70718	97346
potted plant	0.01	0.15	0.01	0.07	42	63	344	41028
bed	0.00	0.05	0.02	0.13	35400	35403	38503	104834
dining table	0.00	0.03	0.01	0.08	2086	2518	4904	155814
toilet	0.00	0.00	0.06	0.46	6487	6488	7587	55453
tv	0.01	0.01	0.06	0.13	4051	5392	5980	13694
laptop	0.00	0.01	0.03	0.27	4873	7527	8499	53879
mouse	0.00	0.00	0.00	0.01	2931	3182	3945	11788
remote	0.00	0.00	0.01	0.02	582	1059	1451	5206
keyboard	0.00	0.00	0.02	0.10	1598	1641	2370	16518
cell phone	0.03	0.03	0.06	0.08	271	411	7511	54850
microwave	0.00	0.00	0.03	0.10	1145	1552	2132	6753
oven	0.00	0.01	0.01	0.06	2209	2481	3108	9534
toaster	0.00	0.00	0.00	0.00	313	355	671	6128
sink	0.02	0.17	0.02	0.24	7499	7504	10872	23125
refrigerator	0.03	0.20	0.06	0.29	2668	4538	5539	9489
book	0.00	0.03	0.00	0.06	91334	91370	100048	614830
clock	0.00	0.01	0.05	0.21	18514	19767	21643	88394
vase	0.00	0.01	0.05	0.14	6415	6415	9596	17080
scissors	0.00	0.00	0.03	0.05	443	460	643	4789
teddy bear	0.00	0.00	0.05	0.17	1856	2273	6276	13012
hair drier	0.00	0.00	0.00	0.00	0	80	132	1031

Continued on next page

	zero shot mAP	zero shot recall	dedicated data mAP	dedicated data recall	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
toothbrush	0.00	0.00	0.00	0.00	1090	1222	1450	10523

Table 7.1: Per-dataset mAP and recall (Left) and Frequency of COCO classes in 3.9M dataset (Right)

7.8 SEGINW sample frequency

Dataset	Category	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
Elephants	elephant	5677	5916	7352	13497
Hand-Metal	hand	57766	59320	78486	117690
	metal	4386	4508	4801	12211
Watermelon	watermelon	1276	1277	1497	29344
House-Parts	aluminium door	3	19	32	23575
	aluminium window	9	33	79	39695
	cellar window	0	0	0	37282
	mint cond roof	0	0	0	0
	plaster	90	95	111	284
	plastic door	2	2	4	21305
	plastic window	6	6	17	44488
	plate fascade	0	0	0	1
	wooden door	215	309	17022	33494
	wooden fascade	0	0	0	175
	wooden window	27	47	98	41071
	worn cond roof	0	0	0	185
HouseHold-Items	bottle	85940	85940	108696	149668
	mouse	2931	3182	3945	11788
	perfume	5869	5869	5992	8949
	phone	8768	9124	14668	116742
Strawberry	R_strawberry	0	0	15245	20877
	people	25278	25278	30665	281076
Fruits	apple	12637	13569	15136	43329
	lemon	7137	7599	8544	17619
	orange	7574	7945	8107	10397
	pear	916	1070	1448	23603
	strawberry	15222	16053	19191	24807
Nutterfly-Squireel	butterfly	26335	26650	30470	38557
	squirrel	897	980	1130	5329
Hand	Hand-Segmentation	0	0	0	8
	hand	57766	59320	78486	117690
Garbage	bin	363	363	632	4549
	garbage	631	643	798	2569
	pavement	123	123	127	157
	road	10289	10892	11327	20860
Chicken	chicken	4856	5136	5513	10394
Rail	rail	767	805	1018	2911
Airplane-Parts	Airplane	5158	9746	11529	21470
	Body	3472	3474	3884	5325
	Cockpit	85	93	117	196
	Engine	3701	3935	4510	9222
	Wing	1297	1297	1514	4645
Brain-Tumor	tumor	82	82	94	124

Continued on next page

Dataset	Category	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
Poles	poles	140	171	216	1594
Electric-Shaver	caorau	0	0	0	9
Bottles	bottle	85940	85940	108696	149668
	can	698	716	1215	3454
	label	2185	2751	2952	17714
Toolkits	Allen-key	0	0	1	1733
	block	4151	4152	5274	20680
	gasket	33	61	144	793
	plier	3	167	224	1393
	prism	89	94	108	190
	screw	151	157	553	8775
	screwdriver	216	238	412	2100
	wrench	398	500	622	6883
Trash	Aluminium foil	6	52	91	2324
	Cigarette	182	236	513	8847
	Clear plastic bottle	1	16	304	48599
	Corrugated carton	0	0	61	2440
	Disposable plastic cup	1	14	285	2146
	Drink Can	9	9	63	40128
	Egg Carton	5	5	14	5566
	Foam cup	2	7	10	1343
	Food Can	1	1	48	4083
	Garbage bag	63	117	183	2355
	Glass bottle	1354	1702	3323	106102
	Glass cup	444	760	1417	66436
	Metal bottle cap	6	7	92	626
	Other carton	0	0	0	23505
	Other plastic bottle	0	0	288	58102
	Paper cup	231	335	481	28889
	Plastic bag - wrapper	0	0	191	1489
	Plastic bottle cap	5	9	103	1348
	Plastic lid	7	12	31	5180
	Plastic straw	20	32	282	1673
	Pop tab	0	0	0	3
	Styrofoam piece	0	0	12	44
Salmon-Fillet	Salmon_fillet	0	12	44	1791
Puppies	puppy	408	736	1189	46169
Tablets	tablets	220	220	2626	7206
Phones	phone	8768	9124	14668	116742
Cows	cow	939	1240	1533	38581
Ginger-Garlic	garlic	972	1048	1170	2707
	ginger	423	435	466	925

Table 7.2: Occurrences of SEGINW Classes in the 3.9M Dataset

7.9 COCO classes sample frequency in dedicated data

	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
person	3313	3313	3610	4872
bicycle	4628	6189	11391	24962
car	21360	21360	23280	34481
motorcycle	4163	4841	7323	23098
airplane	1651	3439	5379	9569
bus	3777	3958	4696	7616
train	4443	4444	5518	12180
truck	3247	3248	3984	29676
boat	6318	6337	7944	15049
traffic light	3202	4309	6926	11971
fire hydrant	649	2736	4285	7379
stop sign	467	607	1108	6380
parking meter	1212	2727	5910	9225
bench	5514	5514	9072	16282
bird	5401	5402	6411	9698
cat	4680	4682	6375	8080
dog	6887	6888	8778	12324
horse	7545	7546	11348	16958
sheep	8163	10033	12873	21163
cow	2105	2769	3694	28449
elephant	3869	4202	5784	10388
bear	4475	4476	10458	25446
zebra	2973	3383	4441	8089
giraffe	2334	2751	3739	8744
backpack	2385	2570	4350	17326
umbrella	3677	3677	5267	8542
handbag	1295	1680	12176	29517
tie	2858	3854	5999	10521
suitcase	2784	3091	5921	16225
frisbee	1124	1259	1693	2916
skis	1801	4402	14800	28105
snowboard	3578	4111	7407	43962
sports ball	3	7	326	11843
kite	4832	6115	7848	9897
baseball bat	881	1274	2976	12411
baseball glove	1231	1466	2449	11230
skateboard	4507	5979	10642	36580
surfboard	5961	8358	13755	32836
tennis racket	1090	1989	3214	11715
bottle	5053	5054	6826	14962
wine glass	2076	3058	5139	15121
cup	5652	5653	8849	17624
fork	4856	6236	9362	17033

Continued on next page

	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
knife	5153	5153	8711	23128
spoon	7314	7316	11630	25162
bowl	10426	10431	14236	24197
banana	915	982	1878	2911
apple	4195	5035	6110	9926
sandwich	7051	8468	11631	17958
orange	1978	2222	2327	5106
broccoli	2572	3316	4669	8384
carrot	3924	3934	6715	8787
hot dog	3778	6725	9650	13103
pizza	7720	7720	9367	14252
donut	3089	4829	8377	14328
cake	10702	10722	16736	25334
chair	12174	12175	15618	27622
couch	3024	4886	10089	14874
potted plant	43	69	499	14771
bed	7492	7494	8581	17502
dining table	157	274	775	22663
toilet	3205	3206	3936	12323
tv	1360	1837	2255	4664
laptop	2200	4078	4795	13528
mouse	3684	4133	4974	9293
remote	1984	3455	4389	6159
keyboard	1864	2272	3390	8651
cell phone	34	74	1581	4798
microwave	3068	4692	7678	18141
oven	1406	1676	2917	20871
toaster	4219	5011	9388	22660
sink	5094	5094	8140	16245
refrigerator	2111	4702	6313	10044
book	7386	7387	8281	22930
clock	5620	6497	8265	18092
vase	7842	7845	12886	22670
scissors	8969	9162	13435	26146
teddy bear	4382	7983	20367	31024
hair drier	0	661	1097	2208
toothbrush	3332	4002	5345	11538

Table 7.3: Occurrences of COCO Classes in the dedicate dataset

7.10 SEGINW classes sample frequency in dedicated data

Dataset	Category	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
Elephants	elephant	3869	4202	5784	10388
Hand-Metal	hand	8050	8642	11462	19772
	metal	1184	1201	1306	2239
Watermelon	watermelon	21	22	38	6549
House-Parts	aluminium door	0	0	0	1516
	aluminium window	0	1	1	2038
	cellar window	0	0	0	2008
	mint cond roof	0	0	0	0
	plaster	1	1	2	17
	plastic door	0	0	0	1424
	plastic window	0	0	0	2666
	plate fascade	0	0	0	0
	wooden door	1	5	1198	2289
	wooden fascade	0	0	0	33
	wooden window	0	0	0	2153
	worn cond roof	0	0	0	0
HouseHold-Items	bottle	10106	10108	13652	29924
	mouse	3684	4133	4974	9293
	perfume	78	78	89	167
	phone	1614	1752	2744	9558
Strawberry	R_strawberry	0	0	1324	1929
	people	486	486	542	3978
Fruits	apple	4195	5035	6110	9926
	lemon	469	497	730	2818
	orange	1978	2222	2327	5106
	pear	53	85	141	5512
	strawberry	1324	1487	1704	2280
Nutterfly-Squireel	butterfly	470	500	670	894
	squirrel	10	10	15	163
Hand	Hand-Segmentation	0	0	0	0
	hand	8050	8642	11462	19772
Garbage	bin	18	18	20	189
	garbage	13	17	22	63
	pavement	55	55	83	102
	road	1592	1663	1695	4706
Chicken	chicken	89	95	114	334
Rail	rail	242	247	317	1936
Airplane-Parts	Airplane	1651	3439	5379	9569
	Body	78	78	83	101
	Cockpit	154	185	203	232
	Engine	648	728	1052	1900
	Wing	472	472	501	947

Continued on next page

Dataset	Category	$n_{0.99}$	$n_{0.95}$	$n_{0.9}$	$n_{0.8}$
Brain-Tumor	tumor	0	0	0	0
Poles	poles	710	791	913	1513
Electric-Shaver	caorau	0	0	0	0
Bottles	bottle	10106	10108	13652	29924
	can	19	19	31	65
	label	57	79	81	233
Toolkits	Allen-key	0	0	0	230
	block	85	85	92	7083
	gasket	0	0	0	6
	plier	0	16	23	9346
	prism	1	1	2	6
	screw	3	4	13	688
	screwdriver	4	4	25	47
	wrench	6	7	9	740
Trash	Aluminium foil	0	3	3	125
	Cigarette	4	4	7	310
	Clear plastic bottle	0	13	152	8445
	Corrugated carton	0	0	3	98
	Disposable plastic cup	0	1	12	563
	Drink Can	0	0	0	1744
	Egg Carton	0	0	0	220
	Foam cup	0	0	0	95
	Food Can	0	0	0	77
	Garbage bag	0	0	0	49
	Glass bottle	1171	1469	2661	24475
	Glass cup	206	299	537	24655
	Metal bottle cap	0	0	9	86
	Other carton	0	0	0	637
	Other plastic bottle	0	0	102	9584
	Paper cup	37	56	103	8454
	Plastic bag - wrapper	0	0	8	38
	Plastic bottle cap	0	0	20	327
	Plastic lid	23	33	52	848
	Plastic straw	0	0	0	45
	Pop tab	0	0	0	0
	Styrofoam piece	0	0	2	2
Salmon-Fillet	Salmon_fillet	0	0	0	75
Puppies	puppy	166	319	492	8571
Tablets	tablets	0	0	92	434
Phones	phone	1614	1752	2744	9558
Cows	cow	2105	2769	3694	28449
Ginger-Garlic	garlic	93	115	125	304
	ginger	28	28	30	73

Table 7.4: Occurrences of SEGINW Classes in the dedicated dataset

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

ENHANCED OBJECT PROPOSAL GENERATION FOR WEAKLY SUPERVISED INSTANCE SEGMENTATION

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
UMAR	KARIM FRISO

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZURICH, 13.09.2023	K. Umar

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.