# MINIMALLY SUPERVISED TRACKING OF ANIMAL COLONIES WITH ITERATIVELY TRAINED OBJECT DETECTORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OĞUZ GÖDELEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JANUARY 2025

Approval of the thesis:

**MINIMALLY SUPERVISED TRACKING OF ANIMAL COLONIES WITH ITERATIVELY TRAINED OBJECT DETECTORS**

submitted by **OĞUZ GÖDELEK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Hande Alemdar
Supervisor, **Computer Engineering, METU** _____

**Examining Committee Members:**

Prof. Dr. Ahmet Burak Can
Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

Assoc. Prof. Dr. Ramazan Gökberk Cinbiş
Computer Engineering, METU _____

Date:09.01.2025

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Surname: Oğuz Gödelek

Signature :

# ABSTRACT

## MINIMALLY SUPERVISED TRACKING OF ANIMAL COLONIES WITH ITERATIVELY TRAINED OBJECT DETECTORS

Gödelek, Oğuz
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Hande Alemdar

January 2025, 58 pages

Animal colonies exhibit highly intricate behaviours, many of which remain poorly understood or unexplored. Effectively monitoring these behaviours requires long-term tracking of a substantial proportion of the group members in their natural environments or an experimental setup. Recent advances in computer vision indicate that neural networks can reliably detect individuals within an animal colony, even in challenging environmental conditions. However, training a neural network with an error rate acceptable for scientific purposes generally requires a large amount of human-labeled training data. In this thesis, we propose an individual animal tracking framework without requiring any explicit human annotations by modifying one of the oldest semi-supervised learning methods called self-training with significant upgrades. Replacing the initial human-annotated dataset required for self-training with the unreliable object locations proposed by the Segment Anything Model (SAM), we iteratively train accurate object detectors. To demonstrate the effectiveness of our method, we conduct some comparative experiments containing our honeybee colony data and a few publicly available animal colony location datasets. The experimental results show that the object detectors trained with the proposed method can achieve

scientifically satisfactory detection results without labelling any bounding boxes.

Keywords: animal tracking, computer vision, high-density object detection, self-training, knowledge-distillation.

# ÖZ

## YİNELEMELİ EĞİTİLEN NESNE ALGILAYICILAR İLE HAYVAN KOLONİLERİNİN MİNİMAL DENETİMLİ TAKİBİ

Gödelek, Oğuz
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Hande Alemdar

Ocak 2025 , 58 sayfa

Hayvan kolonileri, birçoğu yeterince anlaşılmamış veya keşfedilmemiş olan son derece karmaşık davranışlar sergiler. Bu davranışların etkili bir şekilde belirlenmesi, grup üyelerinin önemli bir kısmının doğal ortamlarında veya deneysel düzeneklerde uzun süreli takip edilmesini gerektirir. Bilgisayarla görü alanındaki son gelişmeler, derin sinir ağlarının zorlu çevre koşullarında bile bir hayvan kolonisindeki bireyleri güvenilir bir şekilde tespit edebileceğini göstermektedir. Bununla birlikte, bilimsel amaçlar için kabul edilebilir bir hata oranına sahip bir sinir ağının eğitilmesi genellikle büyük miktarda insan etiketli eğitim verisi gerektirir. Bu tezde, kendi kendine eğitim olarak adlandırılan en eski yarı denetimli öğrenme yöntemlerinden birini Her Şeyi Bölütle (SAM) isimli yeni bir model ile destekleyerek herhangi bir işaretli veriye ihtiyaç duymayan bir hayvan takibi sistemi öneriyoruz. Kendi kendine öğrenme için ihtiyaç duyulan ilk veri setini SAM tarafından önerilen konumlar ile değiştirerek yüksek doğruluklu obje tespit eden modelleri yinelemeli olarak eğitiyoruz. Yöntemimizin etkinliğini göstermek için, bal arısı kolonisi verilerimizi ve halka açık birkaç hayvan kolonisi tespiti veri kümesini içeren bazı karşılaştırmalı deneyler gerçekleştiriyoruz.

Deneysel sonuçlar, önerilen yöntemle elde edilen nesne dedektörlerinin, herhangi bir veri işaretlemeden bilimsel olarak tatmin edici sonuçlara ulaşabileceğini gösteriyor.

Anahtar Kelimeler: hayvan takibi, bilgisayarlı görü, yüksek yoğunluklu obje tespiti, kendine eğitim, bilgi damıtma.

To my lovely wife and family.

## ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Assoc. Prof. Dr. Hande Alemdar, for her invaluable guidance, support, and encouragement throughout the development of this thesis.

I am profoundly thankful to the Department of Computer Engineering and Animal Robot Interaction Lab (ARILAB) for providing the resources and environment necessary for conducting this research and to my colleagues and friends, who have shared this journey with me.

To my family, I am forever grateful for their love and belief in me. Their unwavering support throughout my academic journey has been my cornerstone since I was a child.

I owe a special debt of gratitude to my wife, Hanife, who has been my primary study mate throughout my time at METU. I am thankful for her encouragement, love and spirit.

This work was partially supported by the Scientific and Technological Research Council of Turkey (TÜBITAK) through the project titled “Kovan 4.0: Bal Arısı (Apis Mellifera) Izleminde Yapay Zeka Temelli Nesnel Yaklasımlar” with project reference 122E014 and by EU-H2020-FET-Open (RIA) through the project titled "Robotic Replicants for Optimizing the Yield by Augmenting Living Ecosystem" with project reference 964492.

# TABLE OF CONTENTS

## LIST OF TABLES

TABLES

# LIST OF FIGURES

FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| CV | Computer Vision |
| SAM | Segment Anything Model |
| NLP | Natural Language Processing |
| IoU | Intersection over Union |
| AP | Average Precision |
| CNN | Convolutional Neural Networks |
| MOTA | Multiple Object Tracking Accuracy |
| MOTP | Multiple Object Tracking Precision |
| 2D | 2 Dimensional |
| GUI | Graphical User Interface |
| SVMs | Support Vector Machines |
| GPS | Global Positioning System |
| RFID | Radio Frequency Identification |
| ViT | Vision Transformer |
| FPS | Frame Per Second |

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation and Problem Definition

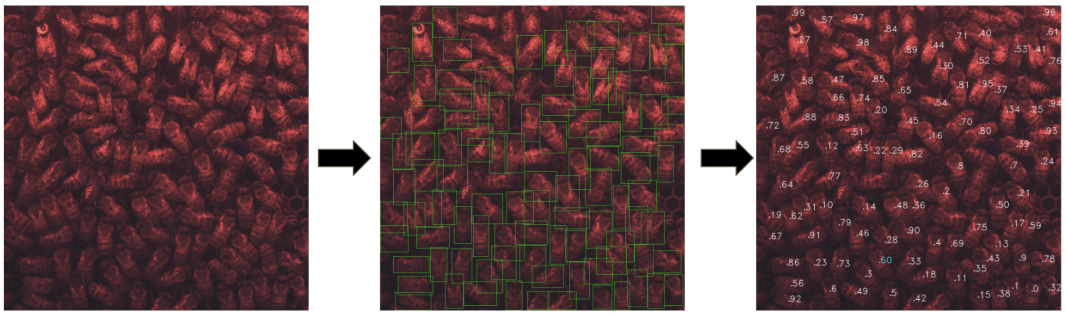


Figure 1.1: Problem definition through single image to bounding boxes and track identities.

The monitoring of animal colonies is fundamental for understanding the dynamics of the ecosystem, which helps humans to evaluate animal behaviours and propose effective conservation and agriculture strategies. It provides deep insights into social organization, movement patterns and behavioural changes of animals, which are vital for assessing the health of the population and the effects of environmental changes.

Recent successes starting with using neural networks in computer vision have transformed the domain by providing accurate tracking of the individuals. Contrary to traditional research approaches mostly based on human observations or predefined image processing techniques, which are both error-prone and labour-heavy, deep learning models can identify individual animals even in complex and dynamic environments. Furthermore, classical techniques are constrained by the necessity of precisely restricted experimental setups or attached markers, which preclude their ap-

plication in monitoring animal colonies in their natural environments. On the other hand, deep learning-based computer vision methods operate independently of these restrictions, allowing perfect study of the behavioural study of animal colonies in wildlife.

Despite their excellent animal detection performance of neural network-based techniques, reaching an acceptable error rate often necessitates vast amounts of human-labelled training data, which are high-resolution images collected from wildlife of up to a few hundred annotated bounding boxes. Annotating even a single image can be a time-intensive process, requiring significant expertise and effort.

## 1.2 Proposed Methods and Models

This study first presents a novel iterative algorithm for training neural network-based object detectors without using explicitly annotated training data. To train an object detector, we enhance one of the oldest and most well-known semi-supervised learning methods called self-training with a recent Segment Anything Model. Initially, a minimal set of annotated data is generated by utilizing and filtering the segmentation maps proposed by SAM [1], which initiates the iterative training process. In each iteration, the previously trained detector is applied to a randomly selected subset of the unlabeled data, thereby expanding the annotated dataset. Then, a new detector is trained from scratch to be used in the following iteration. Through each iteration of the refinement process of the object detector, our deep learning models improve their detection capabilities. Then, we develop a fast object tracker to correctly monitor the assigned identities of individual animals which uses tracking by detection strategy with the help of Hungarian Matching and 2D Kalman Filter.

## 1.3 Contributions and Novelties

Our contributions are as follows:

- We propose a new iterative method to train object detectors effectively without using human-annotated training data.

- We investigate the hyper-parameter selection and its effects on the results of our algorithms.
- We present an end-to-end animal colony tracking framework to monitor the position and identities of individual animals.

All codes are publicly available on https://github.com/oguzgodelek/Minimally-Supervised-Tracking-of-Animal-Colonies-with-Iteratively-Trained-Object-Detectors.

## 1.4 The Outline of the Thesis

The organization of the thesis as follows:

Chapter 2 presents a comprehensive review of current animal tracking systems and their performance. Additionally, we provide background information on pseudo-label based semi-supervised object detection and SAM.

Chapter 3 proposes a novel iterative algorithm for training effective object detectors. Then, we demonstrate the effectiveness of our detectors in our honeybee dataset and some publicly available object detection datasets.

Chapter 4 includes the tracking of the identity and position of the individuals in experimental video setups. It also contains tracking results using a restricted environmental setup.

Chapter 5 concludes the thesis and suggests future work to improve the tracking accuracy of individual animals.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

In this chapter, we present some related works. This chapter is divided into three subchapters. First, we demonstrate the state-of-the-art animal detection and tracking methods and discuss the challenges in the area. Then, we mention the details of the Segment Anything Model (SAM), which forms the basis of our work. Lastly, we briefly mention pseudo-label based semi-supervised learning approaches in the object detection field, which provides another basis of our work.

## 2.1 Animal Detection and Tracking

Animal tracking research is important in understanding animal behaviour, ecology, and environmental conservation. Researchers try to monitor animal movements and behaviours across vast landscapes, revealing critical data on migration patterns and population dynamics using recent technological devices such as cameras, bio-logging devices and satellite tracking. This tracking information is critical to managing environmental crises like climate change and negative human effects and for developing effective conservation strategies. Furthermore, animal tracking research often reveals previously unknown animal behaviours, which are significant to understanding the dynamics of nature. With the ongoing advancement of technology, animal tracking promises to provide increasingly precise and detailed insights.

Because the scope of the thesis is tracking the animal colonies using images, we skip the tracking methods using non-image data such as GPS collars and RFID sensors. Also, we skip the human observation studies.

The tracking of the animals using images can be investigated into two mainstream approaches.

### 2.1.1 Marker-based Animal Tracking

Marker-based tracking requires the use of physical markers with distinct patterns that can be uniquely determined by computer vision algorithms and used to calculate the position and orientation of the attached objects. For years, they have been used in many daily life applications like camera calibration [2] and ID-aware augmented reality applications [3].

The basic idea behind the creation of the tags is that they possess high-contrast, matchless, unnatural textures to be easily identified by classical image processing techniques like contour detection [4]. Generally, they consist of a code matrix of black and white pixels that each area carries a bit of information and allows determining the identification number of the attached objects.

Marker-based tracking has also been used for animal behavioural studies in recent years due to their precision over long-term tracking. For example, Crall et al. create a 25-bit tag system called BEETag [5] (Behavioural Ecology Tag) to study animal behaviours and locomotion. Their system reserves 15 bits of identification information and 10 bits of error correction. They test their system in experiments with bumblebees (Bombus impatiens) and reported that the system can correctly identify their tags with a precision of more than 99%. Blut et al. [6] use AprilTag [7] based 36-bit barcode to determine the honey bee (*Apis mellifera*) behaviours. Alarcon-Nieto et al. [8] develop an ArUco-based tracking system to be used in populational studies of birds and test their system using zebra finches (*Taeniopygia guttata*).

Although tag systems are used for their accurate re-identification and precise positioning capabilities, they introduce significant challenges to behavioural studies. First, the placement process of the tags requires significant time and labour resources. They have to be attached by experts one by one. Second, they sometimes detach from the animal and detachment means losing track of the individual. Also, they degrade over time and this degradation causes to drop in the detection accuracy. Lastly, the

placement of the tags requires to interfere with the nature of animals. They have to be collected and brought to the laboratory, and a marker has to be attached in the laboratory. Also, tags may cause behavioural changes [9].

### 2.1.2 Markerless Animal Tracking

Markerless animal tracking is a common approach that uses image processing and machine-learning techniques to monitor and determine animal movements and behaviours without placing physical markers or tags. This family of methods minimizes potential stress or behavioural changes by eliminating the requirement for direct interaction with animals. Also, it is more robust to environmental noise and easier to use. Besides these, it allows to provision of more advanced data streaming and analysis such as pose estimation and behaviour detection. These features make them highly suitable for both laboratory and field studies.

In the pre-deep learning era, there were many markerless animal tracking methods developed using classical image processing techniques. For example, Perez-Escudero et al. develops a segmentation-based animal tracker named idTracker [10]. The method is based on background subtraction and blob detection, and tested in zebrafish (*D. rerio*) and medaka fish (*O. latipes*). Fontaine et al. [11] propose a multiple worms (*C. elegans*) tracking framework under significant occlusion using detailed geometric models for worm motion and central difference Kalman Filter. Kimura et al. [12] develops a background subtraction-based honey bee tracker. Although these methods can show some success in tracking targeted animal species, their performance is generally limited and outperformed by machine learning-based techniques.

In recent years, neural network-based animal detectors have attracted attention in the research community since it has been proven that neural network-based models are very successful in computer vision tasks such as object detection [13] and segmentation [14]. For example, Bozek et al. [15] develop a segmentation-based honey bee detection approach with 90% accuracy by modifying a well-known CNN network named UNet [16] with recurrent layers. Pereira et al. develop a multi-animal pose estimation framework called SLEAP [17], which uses various CNN-based neural network architectures. They test their system using mice, flies and honey bee

videos. Also, Mathis et al. develop another continuously evolving animal pose estimation framework called DeepLabCut [18] working with minimal training data (50 - 200 data frames). Over time, it has been proven that it can correctly estimate the pose of several animal species such as horses, fish, mice etc. Besides that, Hebert et.al [19] propose a CNN architecture for pose estimation in nematode worms (*C. elegans*). They trained their networks using synthetic data and achieve a 98% correct pose estimate rate.

The biggest advantage of markerless tracking methods is that they provide a noninvasive tracking approach compared to marker-based approaches. Also, they are easily scalable and adaptable to different species and environmental factors. On the other hand, the training of these models generally requires a significant amount of human-labelled training data, which requires several days of expert effort. Furthermore, alternating environmental conditions can badly affect the tracking performance. Lastly, they generally require a considerable amount of computer resources. These three challenges make it difficult to use computer vision algorithms in animal research. Therefore, data-efficient, easily retrainable and lightweight computer vision systems are crucial for the future of the field.

## 2.2 Segment Anything Model (SAM)

Foundation models represent a groundbreaking innovation in deep learning research. They provide a strong basis for a vast amount of downstream applications. These models are huge transformer-based [20] models pre-trained on massive datasets through self-supervised learning. This pre-training gains them ability to encode comprehensive and generalizable data representations. Foundation models have their greatest impact on NLP field, named Large Language Models, with their strong zero-shot performance. The most prominent examples of Large Language Models are GPT [21] and BERT [22].

Also, there are some other foundation models to serve as comprehensive platforms in other fields, mainly for data generation according to the given text. For example, DALL-E [23] is a foundation model for zero-shot text-to-image generation, Mu-

sicGen [24] is for text-to-music generation, and StarCoder [25] is for text-to-code generation. Also, foundation models can be multi-model. That means that it allows different types of data as input. For example, GPT-4 [26] can take images and text as input.

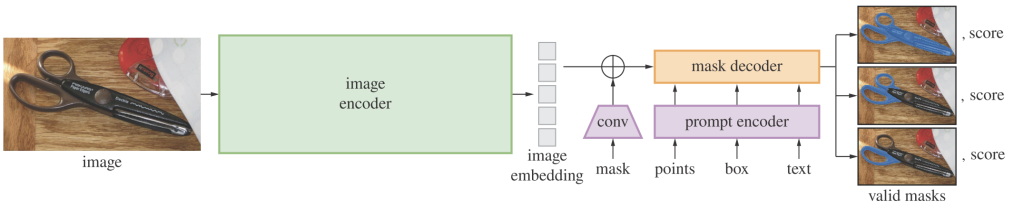



Figure 2.1: SAM overview, taken from [1]

The Segment Anything Model (SAM) [1] is a seminal advance developed by META in the computer vision field. Their goal is to create a foundation model for image segmentation. SAM is also a promptable segmentation model that is able to generate segmentation masks according to text, point or box prompts. It is trained in a massive dataset containing one billion segmentation masks from eleven million images. It possesses a significant zero-shot image segmentation capability, which enables it to be used in several areas including medical images [27] and remote sensing [28]. Also, SAM 2 [29] allows to segment videos using the same architecture with memory attention modules.

Although SAM demonstrates great performance on image segmentation tasks, it possesses some challenges for practical application. First, it requires significant computational resources due to its ViT-based [30] image encoder. To solve the issue, several architectural modifications are proposed to reduce the computational requirements in the literature such as FastSAM [31] and MobileSAM [32]. Furthermore, its capabilities are limited in some applications [33].

## 2.3 Pseudo-label Based Semi-Supervised Object Detection

In recent years, neural networks have greatly succeeded in various domains like object detection [13] and image segmentation [16]. Despite their great success, these models require a vast amount of human-labelled training data, which is a time-consuming

process. However, the semi-supervised learning method is a strong machine learning method proposed to overcome this challenge.

The main idea is to use unlabelled and labelled data while training a neural network by making some assumptions about unlabelled data. The first assumption is the smoothness assumption, which means if two points are close in a high-density region, their outputs are likely to be close to each other. The second assumption is the manifold assumption, which means the points in a higher dimensional space with the same labels lie in the same manifold in a lower dimensional space. The last assumption is the low-density assumption, which states that the decision boundary should pass through a low-density region [34].

Using these three assumptions, many methods have been proposed in the literature. Although many methods use different kinds of pseudo-labelling strategies, there are some methods using other semi-supervised learning strategies such as consistency regularization, whose strategy is to minimize the difference between the output of the different perturbations of the same images [35].

Pseudo-labelling, which is sometimes called as self-training, is one the oldest methodologies in semi-supervised learning history. The main idea is to generate pseudo-labels for unlabelled data and use them during training machine learning methods. More specifically, this technique proposes training an initial learning model. Then, the trained model is applied to the unlabelled data, and pseudo-labels are generated using the initial model. After the selection of created pseudo-labels, a new model is trained with the augmented data set formed by the old data and new data, and this loop continues until convergence. The expectation is that the model refines itself during these self-guided training loops. Many studies prove the effectiveness of self-training in different domains and uses different pseudo-label creation and selection strategies. For example, Rosenberg et al. [36] develop a wavelet transform and search-based object detector. Vandeghen et al. [37] adapts a score histogram-based self-training for object detection using the teacher-student framework. Liu et al. [38] proposes a pseudo-labelling framework for both anchor-free and anchor-based object detectors. Furthermore, Hang et al. [39] develops a soft-label based pseudo-labelling approach.

On the other hand, although they are not actually semi-supervised learning algo-

rithms, it is worth to mention that some algorithms uses weak label strategy to generate pseudo-labels such as [40, 41], which use point prompts to generate pseudo-labels besides the labelled dataset.

# CHAPTER 3

# TRAINING OBJECT DETECTORS VIA ITERATIVE SELF-TRAINING ENHANCED BY SEGMENT ANYTHING MODEL

Object detection is a widely recognized field in computer vision research because it has great potential in several daily and business applications. The basic goal is to determine some rectangular areas on an image that contains interesting objects and their types. Object detectors reach limited success in the pre-deep learning area, although they use complex mathematical models and representations. However, starting from the deep learning revolution, some neural network models like Convolutional Neural Networks (CNN) [42] and Vision Transformers (ViT) [43] significantly exceed the performance of the previous models in terms of detection accuracy and precision.

Although deep learning-based object detectors demonstrate great performance, they have a major drawback. They are primarily trained in a supervised training scheme which requires lots of training images with their bounding box annotations. In an application like individual animal detection, even annotating a single image can be burdensome because some animal colonies like social insects can contain several hundreds or thousands of individuals. The labour requirement of the annotation process limits the usability of these methods.

Furthermore, animal detection is prone to the distribution shift in the data. Different lighting conditions and contamination of the experimental setup can badly affect the incoming images and it causes a performance degradation in object detection. As a result, retraining of the model with the new data is required and new data requires new annotations.

Semi-supervised learning is the approach of reducing the need for data annotations us-

ing labelled and unlabelled data together during training. Generally, semi-supervised learning algorithms use a small subset of annotated data and a huge unlabelled data during training. Mainstream methods exploit the basic assumptions about data like similar data points in higher dimensional space lying on a similar manifold in a lower dimensional space. Although the method demonstrates similar performances with supervised learning-based methods, they also require initially annotated data.

Self-training or self-directed machines is one of the oldest methods in semi-supervised learning. The main idea behind the method is to iteratively learn a model such that it can pseudo-annotate new data at each iteration. In the past, lots of research has proven the correctness of the method on several applications [36, 37].

However, in the default setting, it requires initially annotated data. In our method, we feed the self-training algorithm with the unreliably annotated images by SAM and examine the performance of the method in the animal detection task without labelling any new data. Then, we report the evaluation metrics with different initial data selection methods and self-training configurations.

### 3.1 Initial Dataset Generation Using SAM

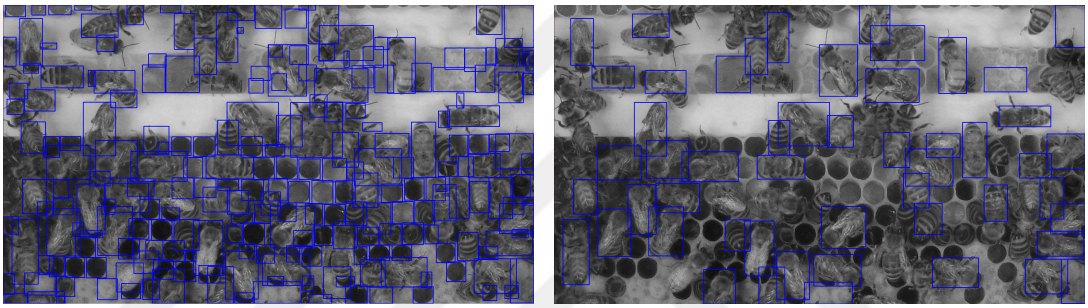
Semi-supervised learning methods leverage both labelled and unlabelled data to learn a prediction function. Self-training is a family of semi-supervised learning approaches that iteratively refine a model. In this method, an initial model is trained on a small set of human-labelled data, and it is used for generating new predictions on unlabelled data to augment the dataset. This iterative process is based on the model's ability to produce accurate pseudo-labels and the assumptions of the smoothness of the data.

The quality of the initial labelled data is important in iterative semi-supervised algorithms because it creates a foundational model to generate reliable pseudo-labels. An initial dataset created carefully ensures that the model begins with robust assumptions of the data, reducing the risk of propagating errors during the iterative pseudo-labeling process. However, weak initial data may cause noisy pseudo-labels, which may lead to poor model convergence over successive iterations. Therefore, the success of iterative methods depends on the quality and representativeness of the initial

human-annotated dataset.

On the other hand, training a strong object detector may be possible with the help of small algorithmic modifications even if the quality of the initial dataset is not perfect. We will give the details of these modifications and the algorithm in Section 3.2.

A non-perfect quality dataset can be generated via a recent computer vision enhancement called vision foundational models. A specific type of vision foundational model named SAM can automatically generate noisy segmentation masks for any given image. These segmentation masks can be converted into object-bounding boxes by simply finding the smallest rectangle that covers the segmentation masks such that the output of the SAM can be used in object detector training.



(a) (b)

Figure 3.1: (a) demonstrates the bounding boxes created by SAM output and (b) demonstrates the bounding boxes of the same image after bounding box elimination.

The problem with using the bounding boxes generated in that way is that SAM generates segmentation masks for all objects and backgrounds in the image when there is no extra input like a prompt or point. To select the relevant bounding boxes, we apply a filtering mechanism that uses basic object features such as the area and aspect ratio of the bounding boxes. Specifically, we eliminate a bounding box if its area is greater than a predefined maximum threshold or if it is lower than a predefined minimum threshold. Also, we apply selection according to the width/height ratio. That means we eliminate a bounding box if its aspect ratio is greater than a predefined maximum threshold or if the inverse of the aspect ratio (height/width) is higher than a predefined minimum threshold. Also, we try to eliminate the bounding boxes according to segmentation confidence, but it does not work because SAM's proposals generally

possess high confidence values.

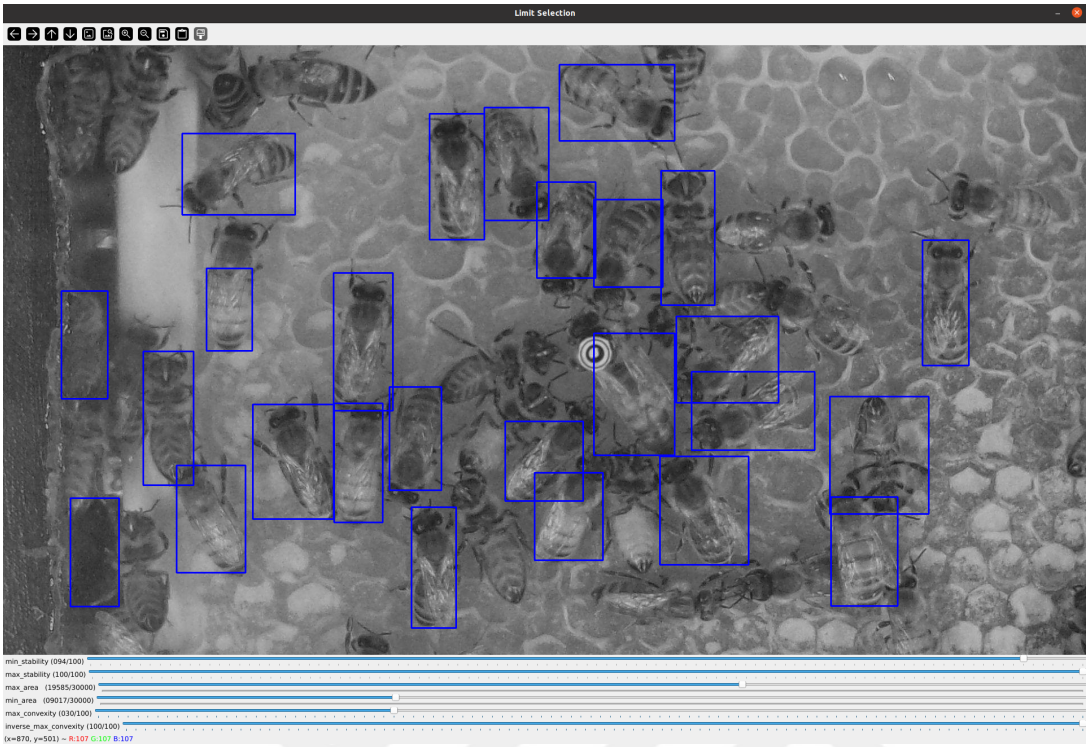



Figure 3.2: The screenshot program created for threshold selection.

To efficiently and quickly determine the threshold values, we generate a basic GUI program 3.2 that takes a sample from the dataset and draws the proposed bounding boxes on the image. It allows changing the threshold values and the image on the screen and updates the bounding boxes drawn on the image. The graphical user interface (GUI) program is designed to prioritize user efficiency and offers a lightweight solution with minimal usage of system resources. Its basic design ensures ease of use, which makes it accessible to users with varying levels of technical expertise. The program features a streamlined layout that allows users to accomplish tasks with minimal effort. Despite its simplicity, the interface provides all necessary functionalities that support required operations without introducing unnecessary complexity. The lightweight nature of the application ensures fast update times of the bounding boxes. We implement the selection GUI using OpenCV [44] user interaction functionality.

By simply observing the effect of the thresholds, the GUI helps users to tune the required threshold parameters rapidly, in even a few minutes. The bar below the image is used for parameter change and the bounding boxes on the images are constantly

updated. Other functionalities of the GUI can be controlled by using the keyboard.

The GUI appears before the SAM generation. When the thresholds seem good enough, after the user presses the start button, the program starts to generate the initial dataset using a random subset of the unlabelled images. The initial generation process is applied to a predefined number of images. The number of images is also a hyperparameter for the self-training iterations. Figure 3.1 demonstrates the effect of bounding box elimination and initial outputs of SAM.

After determining selection criteria, we propose two different strategies for the initial data generation. Our first proposal is to generate a completely random dataset by subsampling the decided number of unlabelled images and generating their annotations with the method described above. Our second proposal uses a little more human supervision. After generating more than the decided number of initial image-annotation pairs using the method described above, the user can select the better-annotated data pairs among the created dataset. This strategy can be helpful for the dataset containing diverse objects or backgrounds, because the output segmentation of the SAM may be poor for some images without any reason, and eliminating bad annotations from the initial dataset can result in a better convergence, even if the detector may have a chance to fix these errors during later self-training iterations.

For the annotation-image pair selection, we provide another easy-to-use GUI named the selection GUI. The selection GUI allows users to select good initial annotations by simply pressing a few buttons. The GUI does not have many functionalities. It basically draws the annotations and moves image-annotation pairs into the accepted or rejected directories according to the user's keyboard input. Selecting the initial dataset using the selection GUI adds only a few minutes to the total processing time.

Furthermore, for the users wanting stronger initial dataset GUI has a easy interface to delete the wrong annotations because initial dataset generation process generally creates more pseudo-labels than the original labels due to its ability to segment all objects. Therefore, in some datasets deleting can be a good option for creating initial dataset.

We conduct experiments using both strategies above using our internal honey bee

dataset. We present the results of these experiments and the effects of the extra human supervision in Section 3.3.

## 3.2 Training Object Detectors via Iterative Algorithm

Training a neural network model using an iterative algorithm is a cyclical process of self-refinement to improve its performance on a given task. The technique is historically called self-training or self-thought learning machine. In this technique, an initial model is trained with small human-labelled data, which forms a foundational model for later iterations. Then, the trained model is applied to the unlabelled data to generate new pseudo-labels for the next iterations. The dataset is augmented with a subset of new pseudo-labelled data and the new model is trained from scratch with the augmented dataset [45].

This process is repeated a few times, and the expectation is that the model iteratively refines itself and converges to a good model state after successive iterations. This paradigm allows the model to use both labelled and unlabelled data such that the need for human annotation can be reduced during machine learning model training.

The main assumption of the technique is that the data points in high-density regions should have identical or similar class labels, which is called the smoothness assumption. Conversely, if two points are separated by a low-density region, they should have different labels.

In the literature, this paradigm is proved in training different types of machine learning models such as linear models [46], SVM's [47], neural networks [48] in different domains and applications like image segmentation [49], object detection [36] etc.

In this section, we propose a version of the self-training algorithm using weak assignments to successively train an object detector. With the help of the process described in 3.1, we generate the initial dataset required for self-training using minimal human supervision. Then, we modify the original algorithm with weak label assignments to correct the artefacts created by the unreliable initial data generation process.

Algorithm 1 demonstrates the pseudocode of the proposed self-training algorithm.

**Algorithm 1** Iterative self-training algorithm enhanced by SAM

**Input:** $X_U$, $h$, $f^{initial}$

$U^0 \leftarrow RandomSample(X_U)$

$L^0 \leftarrow h(U^0)$

$f^0 \leftarrow Train(f^{initial}, U^0, L^0)$

**repeat**

$U^k \leftarrow U^{k-1} \cup RandomSample(X_U)$

$L^k \leftarrow f^{k-1}(U^k)$

$f^k \leftarrow Train(f^{initial}, U^k, L^k)$

**until** convergence or length($X_U$)=0

The algorithm takes an unlabelled image dataset ($X_u$), an initial dataset generation process (h), and an object detector with initial parameters ($f^{initial}$) as input. Then, it randomly samples the dataset and generates pseudo-labels (bounding boxes) for this data via the initial data generation process (SAM segmentations and filtering). Using this initial dataset, we train an initial object detector.

In each iteration, we randomly subsample the unlabelled dataset and increase the number of images in the dataset. By discarding previous pseudo-labels, we create new pseudo-labels for the current dataset (both new images and past images) using the previously trained object detector. Then, we retrain a new detector from the initial weights with the current dataset and generate pseudo-labels. The loop continues until the number of images in the unlabelled dataset is zero.

After each iteration, the model refines its own parameters and converges a parameter state with a better object detection performance. The reason behind the pseudo-label generation of the dataset coming from both past and current data is that we are not confident about the labels created in any iteration. Even in the initial dataset, there are so many artefacts and errors created by SAM. Our method provides a way to fix these artefacts and prevents these errors from propagating through iterations. When the current model becomes confident, its changes to improve these errors increase. Therefore, the model which will be trained in the next loop gets stronger detection performance. The original algorithm of self-training does not require this modification because the initial labels possess the correct labels in the original algorithm.

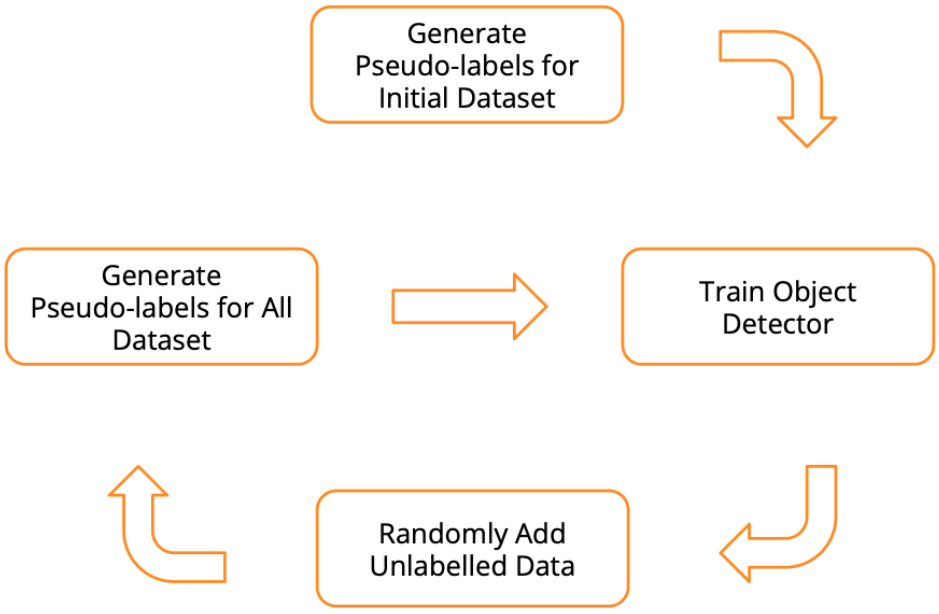



Figure 3.3: Flowchart of the proposed self-training algorithm.

Furthermore, increasing the number of images through iterations serves as a regularizer against error propagation through iterations. To illustrate better, we present the flowchart of our algorithm in Figure 3.3.

On the other hand, this iterative process may result in being stuck in a local poor minimum. However, an object detector is a non-convex model and possesses the possibility to be stuck in a local poor minima regardless of the training and optimisation algorithm. We present our experimental results in Section 3.3, and we observe that the object detectors trained with our algorithm converge good parameters in general despite the chance of being stuck in a poor minimum, when the schedule of new data adding is not aggressive.

## 3.3 Experimental Evaluation

To demonstrate the performance of our model, we conduct detection experiments with three publicly available datasets and one ingroup dataset which consists of honey bee images annotated in our research group. The dataset will be detailed later.

Across all experiments, we utilize YOLOv8.0.145s as our primary object detector, which is a widely recognized object detection model in the literature, and we use Ultralytic implementation [50] and its utility library. Also, we use PyTorch for other tensor operations [51]. YOLOv8s is the second smallest variant within the YOLOv8 family, comprising 11.2 million parameters. The system uses an anchor-free detection approach to better represent the distribution of a custom dataset. This detector was selected due to its efficient resource usage and performance, which are well-suited to the requirements of our application. The model is trained by starting from weights pre-trained on COCO (Common Objects in Context) [52].

The experiment names in the tables below are organized as follows: " Initial Iteration" represents initially trained object detectors that use filtered SAM outputs during their training. "Supervised" represents the model trained with the images, which are the same images in the initially selected dataset, and their human annotations. "$n^{th}$ Iteration" represents the object detector trained with the dataset generated in $n^{th}$ iteration of the self-training.

In all experiments and all self-training iterations, we train the object detector in 20 epochs unless it is specified as a "long" post tag. The experiments and iterations marked as "long" are trained in 100 epochs with the same dataset as its normal version.

During all experiments, we use Adam with weight decay [53] as the optimizer. We use the default parameters of the optimizer suggested by YOLOv8, which are 0.002 as the learning rate, 0.937 as beta1, and 0.0005 as weight decay. We use 3 epochs as the warm-up period and 16 as the batch size.

We utilize strong data augmentations to generalize the different image/lighting conditions and to improve the model robustness during the training of the YOLOv8 object detector. These augmentations cover random colour adjustments like hue, saturation, and brightness alterations to simulate the different environmental conditions. Also, we apply scaling of the images to simulate the objects at different distances from the camera. Also, during training, we use mosaic augmentation, which is the merging of 4 different images into 1 to simulate different scene compositions. Also, we apply random erasing of the input image to improve the robustness of the model.

### 3.3.1 Performance Evaluation

We use AP and maximum F1 score such that we can investigate the final model performance and the improvement over self-training iterations, which will be defined below. Both of them are standard evaluation metrics in the object detection field and are widely recognized. To define these metrics, we should first define 3 pre-metrics which are Intersection over Union (IoU), recall, and precision.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

The output of the object detectors is the bounding boxes that cover the object boundaries and class probabilities. For a single object, it correctly defines a rectangle that encloses the detected object. Intersection over Union (IoU) or Jaccard Similarity is a similarity metric used for determining whether the proposed rectangle is correct or not. It basically calculates the area of the intersection between the proposed bounding box and the ground truth bounding box and the union of these two boxes. The ratio between these two is the IoU metric. In the literature, if IoU is higher than a selected predefined value, then the proposed bounding box is treated as a correct result. During all experiments, we choose the limit of 0.5. That means, if the Jaccard similarity of the proposed bounding box and its ground truth value is higher than 0.5, then it is evaluated as a correct detection.

Precision is the metric of the correctness of the proposed detections. It is basically the proportion of true detections overall detections. On the other hand, recall measures the ability of the model to detect the relevant object in the image. It is defined as the ratio of true detections over all relevant objects. Ideally, the object detectors possess high precision and recall values. However, these two metrics are inversely related to each other, and reporting just these two metrics can be spurious. Therefore, in the object detection literature, a wrapper metric called Average Precision (AP) is often reported. It is basically the area under the Precision-Recall Curve determined by the different confidence thresholds.

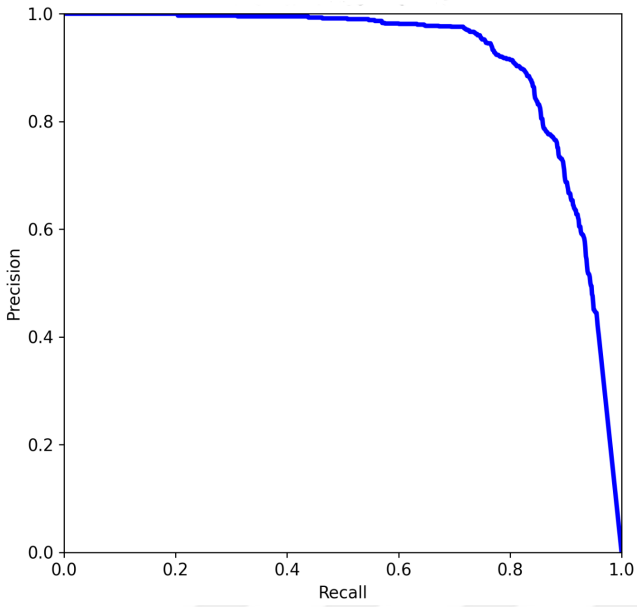



Figure 3.4: Example precision-recall graph from one of our experiments with 0.914 AP value.

$$F_\beta = (1 + \beta^2)\frac{precision.recall}{\beta^2.precision + recall} \tag{3.2}$$

The second metric we use in the performance evaluation is the maximum F1 score. $F_\beta$ score is the metric that measures the predictive performance of a statistical model by weighting recall with $\beta$ against precision. The F1 score is essentially the harmonic mean of the recall and the precision at a selected confidence level. That means recall and precision have equal importance in the metric. The maximum F1 score is the maximum value of the F1 score at any confidence level. It is very suitable to balance the trade-off between recall and precision and confidence level selection for the inference time.

### 3.3.2 Honey Bee Dataset

The honey bee (*Apis mellifera*) dataset consists of images randomly sampled from the internal database of the RoboRoyale project. It contains 1116 unlabelled and 350 labelled high-resolution monochrome bee hive images. We divide labelled images into 2 and we use 150 images as a test dataset and 200 images for training supervised de-

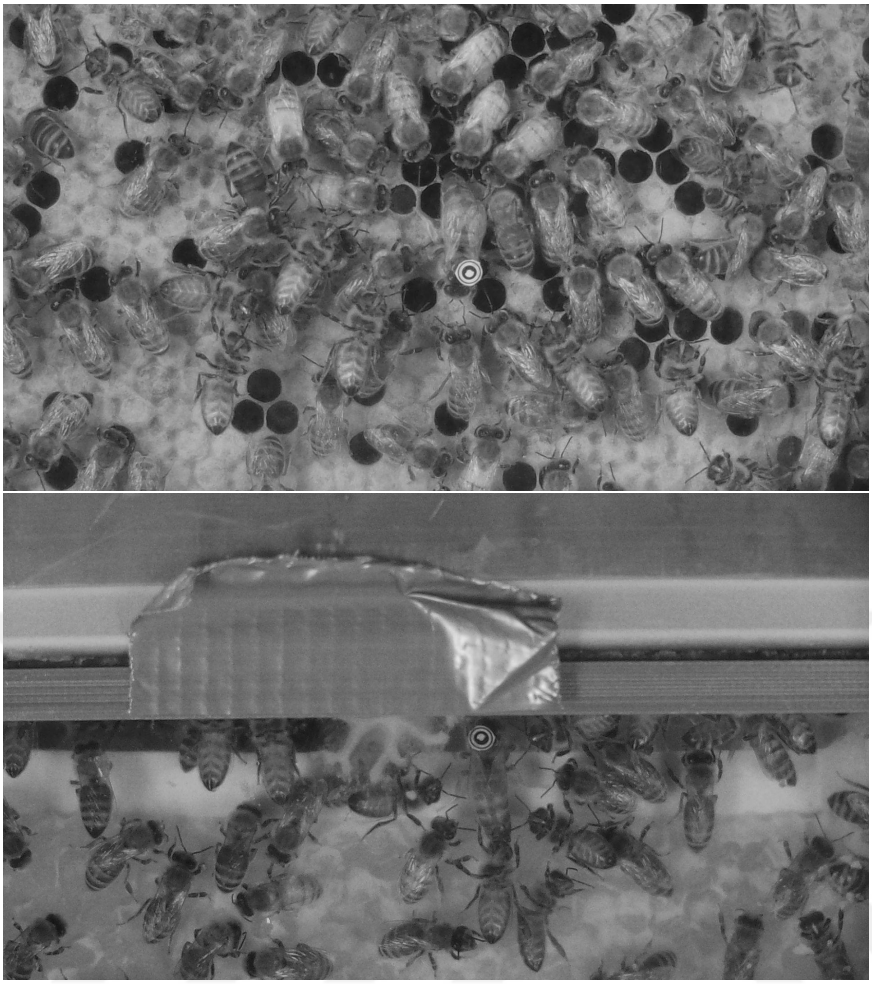


Figure 3.5: Example images from monochrome honey bee detection dataset.

tectors. During the self-training loop, we discard the annotations of these 200 labelled images and we augment the unlabelled dataset with them as if their annotations are not available. All images are taken at the experimental setup at the University of Graz under infrared light. Their resolution is $1920 \times 1080$. The camera in the experimental setup is placed on a robot head and its goal is to track the honey bee queen. Images are labelled in our research group (METU Animal Robot Interaction Lab).

The dataset contains complex backgrounds like comb cells filled with larvae or honey, comb frames and markers indicating the queen bee. Figure 3.5 contains some example images from the dataset. This dataset is our primary dataset because it is an original dataset created in our research group and it includes lots of complex dynamics. There also appears a high-density object arrangement in the dataset. Using the honey

bee dataset, we conduct experiments about the performance difference depending on initial data selection.

To evaluate the correctness and performance of our method, we conduct 3 different experiments on this dataset. In the first experiment, starting from 200 random unlabelled images, we train an object detector using our algorithm. In the second experiment, we reduce the number of images in the initial dataset and observe the performance alterations. In the last experiment, we pseudo-label all unlabelled images and we select initial data using a simple GUI and see the effect of the quality of the initial data on the performance of the algorithm. Even if minimal human supervision in this experiment is involved in the algorithm, the labour cost is much lower than annotating all images.

Table 3.1: Results of our method with 200 random starting images in honeybee dataset

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 200 | 20 | 0.981 | 0.94 |
| Initial Training | 200 | 20 | 0.573 | 0.63 |
| Initial Training Long | 200 | 100 | 0.636 | 0.66 |
| First Iteration | 337 | 20 | 0.815 | 0.73 |
| Second Iteration | 519 | 20 | 0.810 | 0.75 |
| Third Iteration | 808 | 20 | **0.850** | 0.77 |
| Fourth Iteration | 1098 | 20 | 0.847 | **0.78** |

In the first experiment, we randomly select 200 unlabelled images and then use their SAM annotations for the initial dataset. We train the object detector in 4 self-training iterations. In each iteration, the number of annotations generated follows the same ratio. That means 65% of the number of images in the previous dataset are added to the dataset by marking them with the model trained in the previous iteration. Also, the previous images are relabelled by the new model. To determine the performance of our method, we also train a supervised detector using the same 200 images. Table

3.1 demonstrates the performance of the detectors during several iterations. In each iteration, the AP and maximum F1 scores generally increase.

In this experiment, although the supervised training results, which are 0.981 in AP and 0.94 in maximum F1 score, significantly surpass our best results, which are 0.850 in AP and 0.78 in maximum F1 score, the performance of the detector is fair enough to detect some honey bees.

Table 3.2: Results of our method with 20 random starting images in honeybee dataset

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 20 | 20 | 0.792 | 0.74 |
| Supervised | 200 | 20 | 0.981 | 0.94 |
| Initial Iteration | 20 | 20 | 0.486 | 0.53 |
| Initial Iteration Long | 20 | 100 | 0.419 | 0.46 |
| First Iteration | 89 | 20 | 0.663 | 0.66 |
| Second Iteration | 220 | 20 | 0.745 | 0.73 |
| Third Iteration | 451 | 20 | 0.786 | 0.76 |
| Fourth Iteration | 728 | 20 | 0.821 | 0.80 |
| Fifth Iteration | 1035 | 20 | 0.857 | 0.82 |
| Sixth Iteration | 1316 | 20 | **0.860** | **0.82** |

Secondly, we investigate the effect of reducing the number of images for the initial dataset. For this experiment, we start our method using 20 random starting images. Table 3.2 demonstrates the per iteration performance of this configuration. Our results show that even though reducing the initial dataset decreases the performance of the early iterations, during later iterations, the model demonstrates slightly better performance than the model trained in the previous experiment. This unexpected result shows that the outputs of SAM exhibit a considerable amount of noise and during self-iteration, the detector model can refine itself correctly. Furthermore, this situation can improve the effectiveness of our method because the inference of the Segment Anything model is costly and the result of this experiment shows that even

20 SAM-generated images are enough to start the self-refinement process.

Table 3.3: Results of our method with 20 selected starting images in honeybee dataset

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 20 | 20 | 0.843 | 0.79 |
| Supervised | 200 | 20 | 0.981 | 0.94 |
| Initial Iteration | 20 | 20 | 0.645 | 0.66 |
| Initial Iteration Long | 20 | 100 | 0.634 | 0.65 |
| First Iteration | 90 | 20 | 0.785 | 0.73 |
| Second Iteration | 222 | 20 | 0.835 | 0.79 |
| Third Iteration | 449 | 20 | 0.895 | 0.84 |
| Fourth Iteration | 724 | 20 | 0.910 | 0.85 |
| Fifth Iteration | 985 | 20 | **0.914** | **0.86** |
| Sixth Iteration | 1315 | 20 | 0.910 | 0.84 |

Lastly, we conduct some experiments to explore the effect of the initial data quality on the performance of the model. During this experiment, we pseudo-labelled 40 images using SAM+postprocessing and selected 20 of them using the selection GUI.

3.3 presents the results of our experiment. Selection of the better-annotated images by SAM improves the AP performance from 0.860 to 0.914 and the F1 score from 0.82 to 0.86. That means, a better initial dataset improves AP by roughly 0.05 points and F1 score 0.04 points. Among all experiments, we observed the best performance results in this configuration, which were 20 selected images.

Using these results we conclude that even though this version slightly improves the performance, the performance gain against the random configuration is negligible and even random generation of the initial dataset is enough to start self-training. In the later iterations, object detectors correctly refine their results.

Against the argument of the fact that the model trained in each iteration does not have an equal number of optimisation steps, we conduct long experiments for all

of our configurations. Object detectors are trained in 100 epochs using an initially generated honey bee dataset and we keep the best parameters of the model by testing an extra validation dataset in case of overfitting. Table 3.3, 3.2 and 3.1 demonstrate the performance of this experiments. These results state that most of the models have already converged in 20 epochs and long training does not significantly improve the final performance of the object detectors. This experiment shows the effectiveness of our algorithm. That means the model can correctly refine its parameters across self-training iterations.

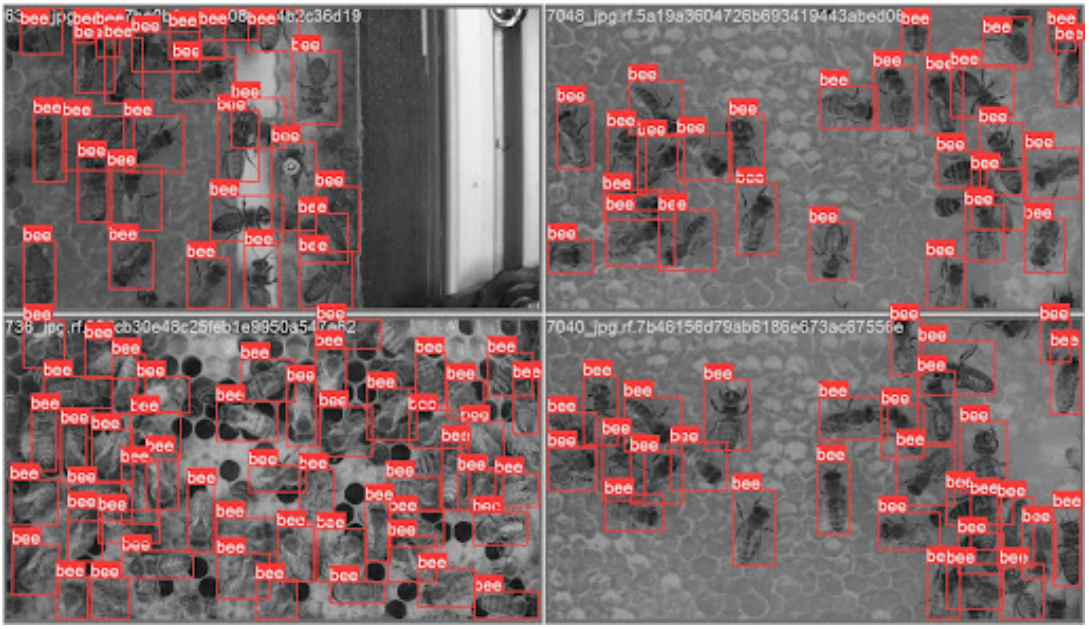


Figure 3.6: Example test images from the honeybee dataset and their proposed outputs of the detector trained in the sixth iteration of our best configuration.

Figure 3.6 presents a quantitative assessment by drawing bounding boxes proposed by the object detector which is trained in the sixth iteration of our best configuration (which is starting from 20 selected images) for 4 random honey comb images in the test dataset. It can be easily said that our strongest honeybee detector can easily find a practically satisfactory amount of honey bees in the complex bee comb images.

Figure 3.7: Example images from aerial sheep detection dataset.

### 3.3.3 Aerial Sheep Dataset and Performance

The aerial sheep dataset is a publicly available dataset published in Roboflow in 2022 [54]. It contains 1727 aerial images. Although most of the images were taken against a grass background, the dataset has its own challenges due to shadows and lighting conditions. Also, the size of the sheep differs in the images because the altitude of the camera is not the same across all the images. The biggest challenge of the dataset is that in some examples the colony gets so close that they almost lose their individual shape. Therefore, individual detection gets harder.

During our experiments, we eliminated some images according to their quality, and we used 1480 of them. We conduct our experiments using 1280 training images without their labels and 200 test images with their labels. All images have the same resolution, and it is 1280×720. In all experiments, we follow the training and test procedure described in Section 3.3.1 and 3.2 except data adding schedule. In this experiment and later experiments, we use doubling data adding schedule which means that the number of new images added to the dataset is equal to the number of images already in the dataset. In the last iteration, we add the remaining unlabelled data to the training dataset. We observe that this data adding schedule provides more stable training in general. Table 3.4 demonstrates the result of the experiments.

Table 3.4: Results of our method with 20 starting images in aerial sheep dataset.

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 20 | 20 | 0.665 | 0.66 |
| Initial Iteration | 20 | 20 | 0.540 | 0.57 |
| First Iteration | 40 | 20 | 0.678 | 0.73 |
| Second Iteration | 80 | 20 | 0.687 | 0.73 |
| Third Iteration | 160 | 20 | 0.706 | 0.74 |
| Fourth Iteration | 320 | 20 | **0.713** | 0.78 |
| Fifth Iteration | 640 | 20 | 0.708 | 0.79 |
| Sixth Iteration | 1280 | 20 | 0.711 | **0.79** |

Across all datasets, the second worst final performance is observed in this dataset because sheep do not have descriptive shapes in the aerial view. That means, that when sheep form a flock, their close proximity to one another makes it challenging to distinguish individual animals.

It can be observed that in each self-iteration the model improves the performance metrics and correctly refines its outputs. Specifically, AP increases from 0.540 to 0.713 after 4 iterations of the new self-training algorithm. The last two iterations do not affect the AP metric, but in the fifth iteration, the maximum F1 score increases by

0.01 points.

Also, Figure 3.8 provides a visual assessment of the performance of our best model, which is the object detector in the sixth iteration. It can be observed that the detector can handle the case which contains sheep with different scales and different lighting conditions. It can even detect most sheep in the cluster, which makes it difficult to identify the individuals.


Figure 3.8: Example test images from sheep dataset and their proposed outputs of the detector trained in the sixth iteration of our best configuration.

### 3.3.4 Aerial Seabird Dataset and Performance

The aerial sea bird dataset is a subset of a huge bird detection dataset which contains over 300,000 bird annotations coming from several different locations [55]. It is a combination of 13 different bird datasets, each of which contains images of different types of birds. The images were taken using unmanned aerial vehicles to explore the nesting behaviour of the birds.

The dataset is characterized by complex backgrounds. It includes land surfaces, glacial areas, sea surfaces, trees, shrubs and swamps, which introduce severe challenges to accurate object detection. This dataset is a valuable resource for evaluating

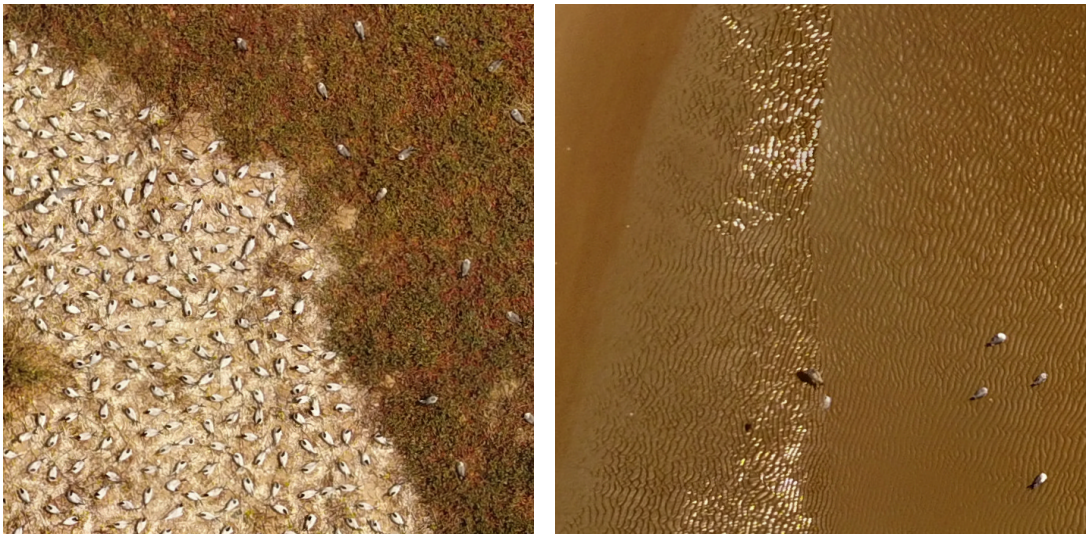


Figure 3.9: Example images from aerial seabird dataset

object detection models because the algorithms have to handle complex daily scenarios. Furthermore, the dataset is important for ecological and conservational studies. Also, the dataset contains highly imbalanced data. In some images, birds are in high-density placement and in some images, there is no bird in the image.

In our experiments, we use a minimal subset of this huge dataset collection such that we can investigate the performance of our algorithm under a low data regime for small object detection tasks because small object detection is a hard problem due to the intrinsic dynamics of convolutional neural networks. On the other hand, this is a common scenario in animal detection since most of the images are taken by drones or planes. We used just one part of the dataset named "terns", which is a tricky part of the dataset. It contains lots of small objects. Figure 3.9 demonstrates sample images from the dataset. Also, the images are highly imbalanced. There are lots of images containing no birds, also there are lots of images containing an outrageous number of birds. In the end, the data we used in the experiments consisted of 280 images in total. We use 110 images and their annotations as test data and 170 images without their annotations as training data.

All training and test samples are low-resolution images, which are $700 \times 700$ RGB images, and we train the object detector as described in Section 3.2 starting from 20 random initial images except we disable erasing augmentation because erasing causes

to disappear lots of objects, and we evaluate them as described in Section 3.3.1. Table 3.5 demonstrates the results of our experiments.

Table 3.5: Results of our method with 20 starting images in aerial sea birds.

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 20 | 20 | 0.659 | 0.70 |
| Initial Iteration | 20 | 20 | 0.321 | 0.45 |
| First Iteration | 40 | 20 | 0.413 | 0.54 |
| Second Iteration | 80 | 20 | 0.491 | 0.61 |
| Third Iteration | 160 | 20 | 0.530 | 0.63 |
| Fourth Iteration | 170 | 20 | **0.542** | **0.65** |

This experiment is our most difficult experiment because the size of the objects is small and we do not have enough data to continue our experiment during lots of iterations. Table 3.5 demonstrates the performance of our experiments. It can be seen that during self-training iterations both metrics constantly improve, which are from 0.321 to 0.542 for AP, from 0.45 to 0.65 for the maximum F1 score.

With these results, even though we can propose that our method is working for small object detection tasks with highly imbalanced datasets, we could not outperform the supervised model just for this dataset. We think that errors during the initial iteration easily propagates through later iterations due to the high number of object in the dataset samples.

Furthermore, in this dataset, there may be no enough sample to correctly refine the parameters of the model because this dataset the smallest dataset we use in the experiments. The detector may reach the performance of the supervised model with extra unlabelled data because the improvement over iterations does not converge yet. However, we could not test this claim since there is no extra data for this dataset.

Figure 3.10: Example images from aerial car detection dataset.

### 3.3.5 Out of Context Experimental Data: Aerial Car Detection

The aerial car dataset [56] is a specialized collection of images captured from aerial perspectives, designed to facilitate research in object detection and tracking tasks. The dataset consists of diverse scenes of environmental variability, which are urban areas, highways and parking lots. Cars within the dataset vary in size, orientation, and appearance, reflecting real-world conditions. It includes different lighting conditions, shadows, and complex backgrounds. This diversity introduces significant challenges for computer vision algorithms. The dataset is an essential resource for traffic monitoring and urban planning applications. Additionally, the dataset is also important for the development of robust detection models capable of operating in dynamic environments like cities. Even if this dataset is out of our experimental context, which is animal detection, we believe that conducting some experiments is beneficial for indicating the generality of our algorithm.

We conduct experiments using 616 training images and 92 test images from the dataset. Different from the first two datasets, the images in this dataset are low-resolution images, whose resolutions are $640 \times 640$. Figure 3.10 represents two sample images from the dataset.

Our experiment starts with 20 random starting images from the dataset, whose labels

Table 3.6: Results of our method with 20 starting images in aerial car dataset.

| Experiment Name | # of images | Epochs | AP | Maximum F1 |
|---|---|---|---|---|
| Supervised | 20 | 20 | 0.867 | 0.80 |
| Initial Iteration | 20 | 20 | 0.675 | 0.67 |
| First Iteration | 40 | 20 | 0.827 | 0.78 |
| Second Iteration | 80 | 20 | 0.833 | 0.81 |
| Third Iteration | 160 | 20 | 0.871 | 0.81 |
| Fourth Iteration | 320 | 20 | 0.884 | 0.82 |
| Fifth Iteration | 616 | 20 | **0.897** | **0.84** |

are generated using the procedure described in Section 3.1. Table 3.6 demonstrates the result of our experiment. The results show that the initially generated images have little noise and that the detector does not show a good detection performance, which is 0.675 AP. Even a single iteration can significantly improve the performance of the model, which is from 0.675 to 0.827.

Then, after each iteration detection model can constantly improve its detection result and it reaches 0.897 AP and 0.84 maximum F1 score at the fifth iteration Figure 3.11 demonstrates some random images from the test dataset and their proposed bounding boxes. Even if there are some erroneous cases like in the above-left image, the final detector can reach a strong state to be used in real-life applications. Also, the model proposes low confidence for these erroneous cases, which the confidence level threshold can easily eliminate.

### 3.3.6 Discussion

In all experiments, our results show that the self-training algorithm shows strong improvement during iteration under the availability of enough unlabelled data. For a more detailed discussion, we divide this chapter into two subchapters. In the first chapter, we provide a summary of our experimental results and in the second chapter,

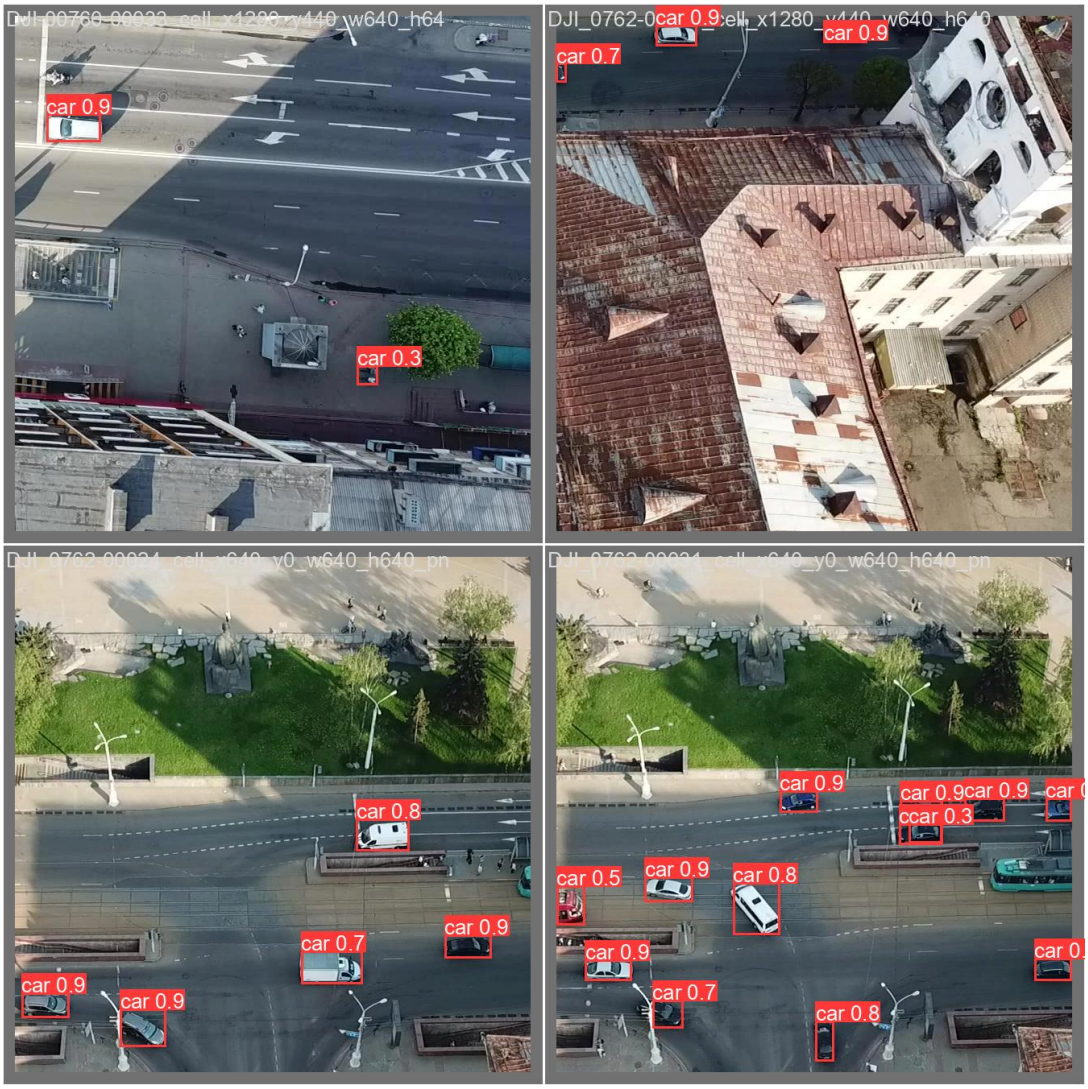
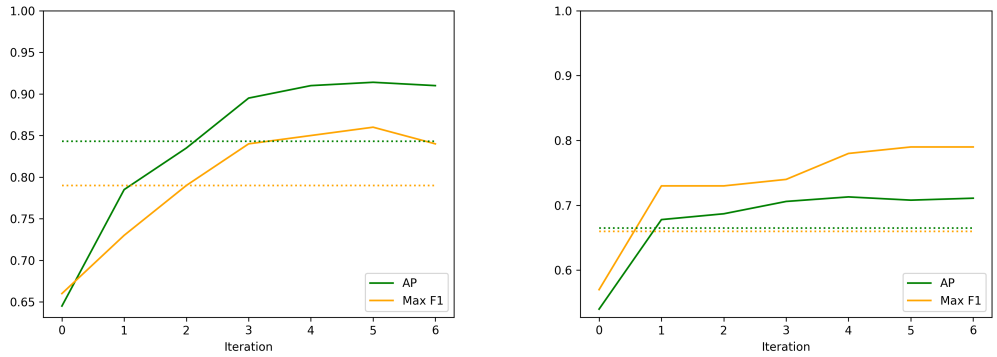



Figure 3.11: Example test images from aerial car dataset and their proposed outputs of the detector trained in the sixth iteration of our best configuration.

we discuss interesting findings about the self-training algorithm.

#### 3.3.6.1 Summary for Experimental Results

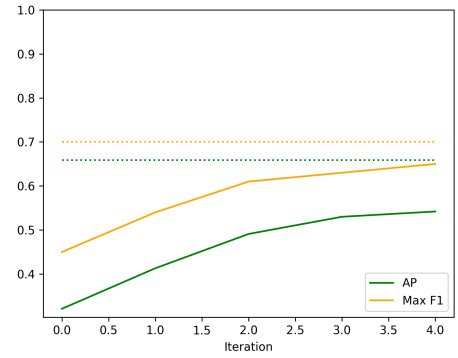
Figure 3.12 presents metric improvements over self-training iterations for four datasets we use in the experiments. The green lines demonstrate AP over iterations and the orange lines demonstrate the maximum F1 score over iterations. Also, the dashed lines are the supervised training results with corresponding colours. For supervised results, we use the same 20 training images we used as the initial dataset.



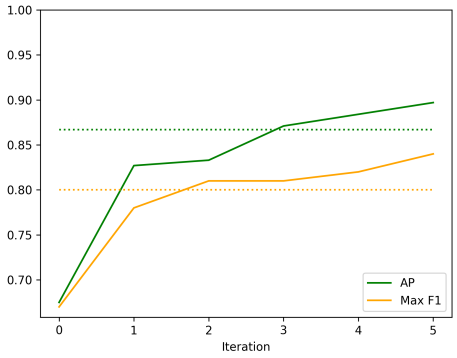

(a) Honeybee




(b) Sheep




(c) Sea Bird




(d) Car

Figure 3.12: Performance improvements over self-training iterations for all datasets.

The above-left plot is the result of our honey bee dataset created by us. The above-right graph belongs to the aerial sheep dataset. The below-left graph shows the results of the high-density seabird dataset and the below-right plot belongs to our out-of-context aerial car detection experiments.

Using these metrics/iterations graphs we can confidently propose that under the availability of enough unlabelled data, our iterative self-training method improves the performance of the object detectors after each iteration for all datasets except seabird due to the lack of enough unlabelled data. In more detail, the improvements during early iterations are more pronounced and strong, whereas the improvements during late iterations are smaller. It seems that during the self-refinement process, object detectors converge after some iterations and their metrics become constant. This situation can

be seen more clearly in honey bee and sheep experiments, whose metrics show small alterations during late iterations. Also, the most confident models trained with our iterative algorithm outperform the model trained in the supervised learning paradigm.

#### 3.3.6.2 Discussion about The Effects on F1-Confidence Curves

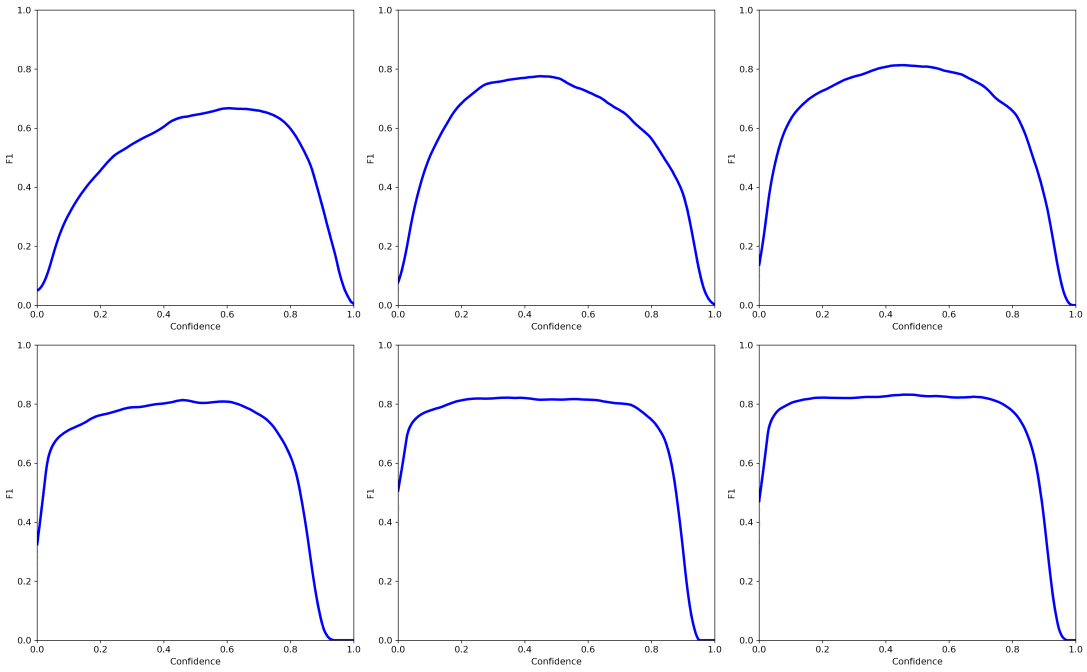



Figure 3.13: F1 - Confidence curves over self-training iterations.

In our experiments, we determine a strange phenomenon that self-training causes over F1-confidence curves. After each self-training iteration, the model possesses a more smooth F1-confidence curve than the previously trained ones. Figure 3.13 demonstrates this phenomenon in the aerial car dataset.

The above-left one is the F1-confidence curve belonging to the object detector trained with the initial dataset, and the below-right one is the F1/confidence curve belonging to the model trained in the fifth iteration. The curves are arranged in ascending order from left to right. This phenomenon can be quantitatively observed in Figure 3.13. Although we present this phenomenon belonging to only one of our experiments, it is observed in all of our experiments.

This quantitative result proposes that our iterative training strategy not only improves

the detection metrics but also makes the model more robust against the selection of confidence threshold because the F1 scores are almost constant around a confidence plateau in later iterations.

This interesting evidence helps to determine the confidence threshold selection because, after each iteration, the detection results of the model are less dependent on the threshold selection. In other words, our algorithm helps to create stronger object detectors over time. It seems that it converts weak object detectors to strong object detectors. It seems that investigating this phenomenon is a promising research direction to create stronger object detection models.

## 3.4 Summary

Tracking of an animal colony generally requires accurate object detection and strong object detectors are trained with excessive amounts of human-annotated training data. On the other hand, labelling even a single image can be burdensome, because an animal colony can contain thousands of individuals. In this chapter, we introduce a self-training variant training algorithm which does not require explicitly annotated images to train an object detector by leveraging self-refinements and SAM. Then, we present the performance of our algorithm in three animal detection datasets and one aerial car dataset to explore the potential application domains. Our findings demonstrate that the algorithm described in this chapter can be a very powerful tool in rapid object detector development and domain adaptation, especially in labour-intensive domains.

# CHAPTER 4

# MARKERLESS ANIMAL COLONY TRACKING BY DETECTION

Object tracking is a critical task in computer vision, which involves identifying the object and monitoring the position of the object across sequential video frames. The most common paradigm begins with the detection of the object, which determines the objects and their positions of interest within a single frame using object detection algorithms such as YOLO [13] and Fast R-CNN [42]. This is followed by track-object association, where objects are linked across frames to maintain consistent trajectories.

In the literature, several algorithms are employed for object tracking. The classical approaches use more mathematical tools like Kalman Filter [57] to predict the next position of the object and use different matching strategies such as Hungarian algorithm [58] and graph-based association [59]. Also, there are some recently proposed methods using neural networks for tracking objects such as MOTR (Multiple Object Tracking with Transformer) [60] and TrackFormer [61].

Object tracking has several applications such as tracking pedestrians and automobiles in autonomous driving, real-time patient monitoring in healthcare, and augmented reality systems where virtual objects must align with real-world movements.

Furthermore, animal tracking is one of the most important application domains of object tracking. Object tracking plays a vital role in animal research. It can offer significant contributions to ecological research, and wildlife conservation strategies. Also, the movement patterns of the animals are valuable insights in behavioural studies.

In this chapter, we propose an animal tracking framework to use the object detectors trained with the iterative algorithm in Chapter 3 and some mathematical predic-

tion and association tools named Kalman Filter and Hungarian Algorithm. Then, we present our experimental results using our restricted honey bee experiment, which aims to determine the effects of well-known pesticides on honey bees.

## 4.1 Tracking Algorithm

To create an animal tracking framework, we adopt the "tracking-by-detection" [62] paradigm. It is a widely used two-stage technique in object-tracking mostly for scenarios which involve dynamic environments. This approach combines object detection and tracking into a single framework, which consecutively detects objects of interest and associates them to the previous tracks across video frames. In this way, decoupling object detection and tracking step, this paradigm allows to improve tracking performance by simply improving only the object detectors or the association method.

For the object detection step, we use an object detector trained with the minimally supervised iterative training algorithm proposed in Chapter 3. In this way, if there is a distribution shift in the tracking data over time due to lighting alterations or environmental changes, we can retrain the object detector without using any human-labelled training data to regain the tracking performance of the model.

As the estimation model, we use the Kalman Filter, and we use the Hungarian algorithm for associating object-track pairs. We observe that this simple and effective approach provides a robust and fast animal tracking framework.

### 4.1.1 Position Estimation Model

In the estimation model, we decide to generate predictions only for the central points of the object. Therefore, the state space of each target is modelled as:

$$\mathbf{x} = [u_x,\, u_y,\, \dot{u}_x,\, \dot{u}_y]^T \tag{4.1}$$

$u_x$ and $u_y$ represent the pixel position of the track's centres, and the dotted versions represent the pixel velocities respectively.

To estimate the new positions and velocities, we use 2D Kalman Filter [57] with the linear motion model, which is proven to be an optimal recursive predictor under Gaussian noise. We use $\frac{1}{FPS}$ as step size which is the time between consecutive frames.

### 4.1.2 Track Association

To associate tracks and object detections, we use Hungarian Algorithm [58], which solves the assignment problem in polynomial time. First, we generate predictions of object locations using the procedure described in Subsection 4.1.1 based on previous locations. Then, we create a distance matrix D whose $D_{ij}$ elements represent L2 distance between the centre coordinates of $i^{th}$ track and $j^{th}$ detection, and we find the minimum cost assignment of the distance matrix D using Hungarian Algorithm. Lastly, we assign the object features to the solution and update the Kalman filter states.

For tracklets whose new detection is not found, we simply update its state using its Kalman Filter predictions and we delete tracks whose detections cannot be found during 10 consecutive frames. Deleting these tracks is helpful for better tracking because the object may leave the screen. Even if it does not leave the screen, the predictions of the linear motion model cannot describe the correct motion track after 10 consecutive frames. For the new detections whose minimum track assignment cannot be found we simply initiate a new track with the detected initial position.

## 4.2 Experimental Results

To demonstrate the effectiveness of our tracker, we conduct some experiments using honey bee experimental data which will be detailed later and present our results in this section.

During all experiments, we use the YOLOv8 [50] object detector and we train it using the iterative procedure. We do not use any test dataset during object detector tracking

### 4.2.1 Performance Evaluation

We use Precision, Recal and MOTA (Multiple Object Tracking Accuracy) from CLEAR MOT [63] metrics to evaluate the tracking performance of the tracker. They are common metrics to evaluate our multi-object tracking performance. We do not use MOTP (Multiple Object Tracking Precision) because it measures the positional precision.

We divide the tracking results into four categories. True positives (TP) are correct detections. When IoU between a tracked bounding box and ground truth bounding box is higher than 0.5, then we accept the detection as correct tracking. False positives (FP) happen when the tracker incorrectly predicts an object where none exists, leading to over-detection. On the other hand, a false negative (FN) arises when the model fails to detect an object that is present, resulting in under-detection. Lastly, an ID switch (IDSW) occurs when two ID assignment switches between two tracks or a new track is assigned to a track which is tracked in the past.

$$Precision = \frac{True\,Positives}{True\,Positives + False\,Positives} \tag{4.2}$$

$$Recall = \frac{True\,Positives}{Ground\,Truths} \tag{4.3}$$

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t GT_t} \tag{4.4}$$

Equation 4.4 shows the component of the MOTA (Multiple Object Tracking Accuracy) metric. It basically represents the overall tracking performance of the detector and tracker demonstrating the erroneous detections caused by false negatives (FN), false positives (FP) and identification switches (IDSW) overall ground truth objects (GT).

### 4.2.2 Restricted Honey Bee Experiment

The restricted honey bee dataset contains a 3000×4000 resolution honey bee video taken by a PointGrey camera. We created this dataset in our research group. For effec-

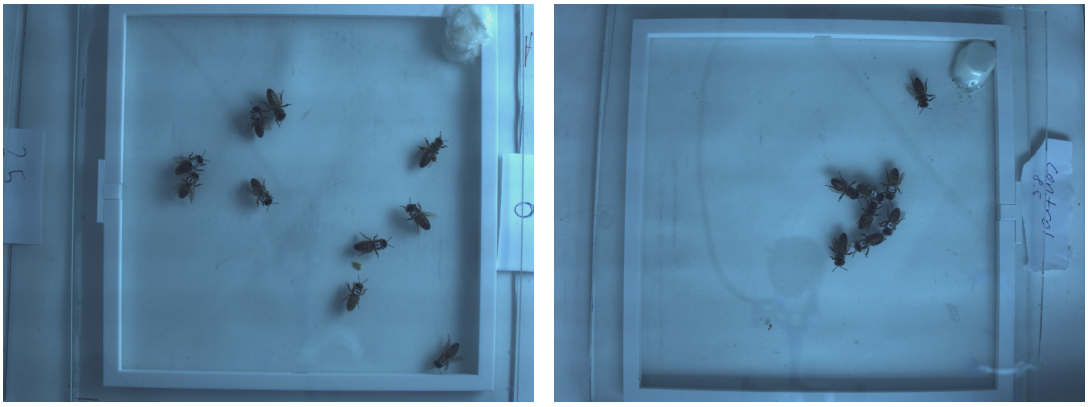


Figure 4.1: Example images from the restricted honey bee experiments

tiveness, the video is downsampled to 750×1000, and the experiments are conducted in this configuration.

The data comes from a set of lithium experiments aiming to determine the effect of lithium on honey bee behaviour. The experimental setup consists of four parallel restricted experiments called arena setup which is a closed area, and every arena contains the same number of honey bees fed by lithium of different doses and each experiment lasts 24 hours. The ultimate goal is to observe the behavioural difference between the honey bees from different lithium doses. The videos are taken at low FPS due to resource limits, and this situation hardens the association between tracks and detection because the position of the bees can rapidly change between consecutive frames.

For the experiments, 672 hours of video are recorded in 4 FPS. To train the object detector, we randomly extract 420 frames from 672 hours 672-hour-long dataset and use 320 of them as unlabelled training data. We label 100 frames to measure the detector's performance after training. The final detector is trained with four iterations of the iterative algorithm and shows **0.918** AP and **0.88** maximum F1 score. We select the confidence threshold as 0.72 which has the maximum F1 score and select the IoU threshold as 0.5.

To test our tracker performance, we randomly select two different videos which contain 10 honey bees, from the dataset, and we apply our tracking framework to the selected videos. All results are reported according to the tracker data.

Table 4.1: Tracking results of our restricted honey bee experiments.

| Experiment Date | # of frames | TP | FP | FN | IDSW | Precision | Recall | MOTA |
|---|---|---|---|---|---|---|---|---|
| 08.05.2024 | 450 | 4355 | 12 | 130 | 3 | 0.99 | 0.97 | 0.97 |
| 03.06.2024 | 480 | 4741 | 34 | 20 | 5 | 0.99 | 0.99 | 0.99 |

Table 4.1 presents our result of tracking experiments. We conduct two different experiments with our framework. The experiments show that our animal tracking framework works exceptionally precisely and accurately in the restricted honey bee experimental setup, and it shows a robust performance across multiple tracking metrics. Also, it can be observed that the prediction and association part can handle the errors coming from the detection part.

The main error source seems false negatives, which are undetected objects. In the test video, there are some shadows due to the field edges. When honeybees are at the edges, the detector can sometimes not detect them. This problem may be solved by adding images with objects in the shadow to the initial unlabelled dataset.

Furthermore, there is a tradeoff between ID switches and false positives in the experiments. We set a threshold for deleting an undetected tracklet over frames. In our settings, if a tracklet cannot be assigned any detection across consecutive 10 frames, then we delete that tracklet. This technique is useful to handle undetected objects because we can assign ID and position for undetected objects due to the predictions of Kalman Filter during a few frames, and when the object is detected again, its motion trajectory is recovered. However, if an object cannot be detected over 10 consecutive frames, then the predictor can generate wrong positional information due to the artefacts in the motion model. When we decrease the deleting threshold, the number of ID switches starts to increase because the systems prefer to assign new IDs for recovered predictions rather than predicting their positions. On the other hand, when we decrease the deleting threshold, the number of falsely predicted positions increases. This tradeoff can be adjusted using the deleting threshold according to the sensitivity of the application.

Also, ID switches are a serious problem for tracking, even if they occur rarely. The

tracking system is designed to be used in long-term tracking experiments. Over long experiments, the count of ID switches will increase and the reliability of the tracker may become questionable. To handle that problem, this system definitely needs a re-ID mechanism, which we will work on in the future.


Figure 4.2: Example tracking images from our test video.

Figure 4.2 shows a frame and its predicted bounding boxes. Overall, although the framework needs some extra work to ensure the reliability of the model for long-term experiments, the tracking framework qualitatively demonstrates an accurate tracking performance.

## 4.3 Summary

In this chapter, we proposed a fast and accurate animal-tracking framework combining an object detector, Kalman filter, and association algorithm to track animals effec-

tively across video sequences. The object detector identifies animals in each frame, while the Kalman filter predicts their positions in subsequent frames, assuming the linear motion dynamics. The association algorithm links detections across frames and maintains consistent identities. To demonstrate the effectiveness of the framework we conduct a series of experiments that contain the same number of honey bees in each frame, and we evaluate the performance of our framework using Precision, Recall and MOTA metrics. Our experimental results show that our tracking framework may significantly help animal studies due to its high performance and fast operations.

# CHAPTER 5

# CONCLUSION AND FUTURE WORKS

Computer vision-aided animal tracking has become a groundbreaking approach in biological research such as wildlife conservation and behavioural studies. They can offer precise information about animal movements and interactions. Using state-of-the-art deep learning-based computer vision models, these systems can provide continuous and non-invasive automated animal monitoring systems, which allows researchers to study previously undiscovered phenomena. Also, they can offer observation data with high spatial and temporal resolution, which classical monitoring systems cannot reach.

Despite their remarkable advantages, computer vision-based animal tracking systems possess notable challenges, particularly regarding data labelling requirements. The performance of these systems heavily relies on large, accurately labelled datasets such that detection models can efficiently train with them. Annotating such datasets is often an error-prone, labour-intensive and time-consuming process because of dealing with complex natural environments where animals may be camouflaged or indistinguishable. Furthermore, alterations in lighting and weather conditions complicate this process more and cause accuracy decline over time since they result in changes in data distribution. Therefore, supervision reduction techniques are crucial for the future of these studies.

In this thesis, first, we propose a minimally supervised animal detector training algorithm requiring only a few minutes to complete the human supervision part. It only requires a selection of bounding box size and aspect ratio using a GUI program. Applying this information, the algorithm generates an initial dataset via the Segment Anything Model for iterative object detector training. In each iteration of the iterative

training, a new object detector is trained with the current data, and the trained detector is applied to the dataset which is the union of past data and randomly added new data. Then, the process repeats until convergence. In our experiments, we observe that during the iterative training process, object detectors can refine their parameters, and after each iteration, their performance improves. Also, we demonstrate that the performance of the models trained with the iterative algorithm shows better performance than the low-their supervised counterparts with low data regimes.

Then, we present an animal tracking framework using only detector predictions and mathematical association frameworks. Our experiments show that the framework demonstrates scientifically satisfactory tracking performance.

We suggest two different future research that seems promising. The first one is to try a deep learning-based object tracker because our method is very simple and even if it correctly tracks the object with linear motion, it may demonstrate a bad performance in dynamic and complex environments. A tracker modification that can learn motion dynamics can be very beneficial to improve tracking accuracy.

The second one is to create a re-ID mechanism using self-supervised learning. Constructing a contrastive self-supervised loss between objects bounded by predicted boxes coming from consecutive frames, an encoder can be trained and used for the identification of objects. This paradigm can be useful, especially for animals with possibly unique identification patterns. Also, it can prove or disprove the possibility of individual identification via learnable functions.

# REFERENCES


[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[2] B. Atcheson, F. Heide, and W. Heidrich, "CALTag: High Precision Fiducial Markers for Camera Calibration," in *Vision, Modeling, and Visualization (2010)* (R. Koch, A. Kolb, and C. Rezk-Salama, eds.), The Eurographics Association, 2010.

[3] J. Rekimoto and Y. Ayatsuka, "Cybercode," *Proceedings of DARE 2000 on Designing augmented reality environments*, april 2000.

[4] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, p. 32–46, april 1985.

[5] J. D. Crall, N. Gravish, A. M. Mountcastle, and S. A. Combes, "Beetag: A low-cost, image-based tracking system for the study of animal behavior and locomotion," *PLOS ONE*, vol. 10, p. e0136487, september 2015.

[6] C. Blut, A. Crespi, D. Mersch, L. Keller, L. Zhao, M. Kollmann, B. Schellscheidt, C. Fülber, and M. Beye, "Automated computer-based detection of encounter behaviours in groups of honeybees," *Scientific Reports*, vol. 7, december 2017.

[7] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[8] G. Alarcón-Nieto, J. M. Graving, J. A. Klarevas-Irby, A. A. Maldonado-Chaparro, I. Mueller, and D. R. Farine, "An automated barcode tracking sys-

tem for behavioural studies in birds," *Methods in Ecology and Evolution*, vol. 9, p. 1536–1547, april 2018.

[9] R. Dennis, R. Newberry, H.-W. Cheng, and I. Estevez, "Appearance matters: Artificial marking alters aggression and stress," *Poultry Science*, vol. 87, p. 1939–1946, october 2008.

[10] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda, and G. G. de Polavieja, "idtracker: tracking individuals in a group by automatic identification of unmarked animals," *Nature Methods*, vol. 11, p. 743–748, june 2014.

[11] E. Fontaine, J. Burdick, and A. Barr, "Automated tracking of multiple c. elegans," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3716–3719, 2006.

[12] T. Kimura, M. Ohashi, K. Crailsheim, T. Schmickl, R. Odaka, and H. Ikeno, "Tracking of multiple honey bees on a flat surface," in *2012 Fifth International Conference on Emerging Trends in Engineering and Technology*, pp. 36–39, IEEE, 2012.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017.

[15] K. Bozek, L. Hebert, Y. Portugal, A. S. Mikheyev, and G. J. Stephens, "Markerless tracking of an entire honey bee colony," *Nature Communications*, vol. 12, march 2021.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[17] T. D. Pereira, N. Tabris, A. Matsliah, D. M. Turner, J. Li, S. Ravindranath, E. S. Papadoyannis, E. Normand, D. S. Deutsch, Z. Y. Wang, G. C. McKenzie-Smith, C. C. Mitelut, M. D. Castro, J. D'Uva, M. Kislin, D. H. Sanes, S. D. Kocher, S. S.-H. Wang, A. L. Falkner, J. W. Shaevitz, and M. Murthy, "Sleap: A deep learning system for multi-animal pose tracking," *Nature Methods*, vol. 19, p. 486–495, april 2022.

[18] J. Lauer, M. Zhou, S. Ye, W. Menegas, S. Schneider, T. Nath, M. M. Rahman, V. Di Santo, D. Soberanes, G. Feng, V. N. Murthy, G. Lauder, C. Dulac, M. W. Mathis, and A. Mathis, "Multi-animal pose estimation, identification and tracking with deeplabcut," *Nature Methods*, vol. 19, p. 496–504, april 2022.

[19] L. Hebert, T. Ahamed, A. C. Costa, L. O'Shaughnessy, and G. J. Stephens, "Wormpose: Image synthesis and convolutional networks for pose estimation in c. elegans," *PLOS Computational Biology*, vol. 17, p. e1008914, april 2021.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.

[22] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[23] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *CoRR*, vol. abs/2102.12092, 2021.

[24] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," 2024.

[25] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor,

J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries, "Starcoder: may the source be with you!," 2023.

[26] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Bel-

bute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "Gpt-4 technical report," 2024.

[27] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, "Segment anything in medical images," *Nature Communications*, vol. 15, january 2024.

[28] S. Ren, F. Luzi, S. Lahrichi, K. Kassaw, L. M. Collins, K. Bradbury, and J. M. Malof, "Segment anything, from space?," 2023.

[29] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[31] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," 2023.

[32] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," 2023.

[33] W. Ji, J. Li, Q. Bi, T. Liu, W. Li, and L. Cheng, "Segment anything is not always perfect: An investigation of sam on different real-world applications," *Machine Intelligence Research*, april 2024.

[34] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, p. 373–440, november 2019.

[35] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[36] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, vol. 1, pp. 29–36, 2005.

[37] R. Vandeghen, G. Louppe, and M. V. Droogenbroeck, "Adaptive self-training for object detection," 2023.

[38] Y.-C. Liu, C.-Y. Ma, and Z. Kira, "Unbiased teacher v2: Semi-supervised object detection for anchor-free and anchor-based detectors," 2022.

[39] Y. Tang, W. Chen, Y. Luo, and Y. Zhang, "Humble teachers teach better students for semi-supervised object detection," *CoRR*, vol. abs/2106.10456, 2021.

[40] L. Chen, T. Yang, X. Zhang, W. Zhang, and J. Sun, "Points as queries: Weakly semi-supervised object detection by points," *CoRR*, vol. abs/2104.07434, 2021.

[41] S. Zhang, Z. Yu, L. Liu, X. Wang, A. Zhou, and K. Chen, "Group r-cnn for weakly semi-supervised object detection with points," 2022.

[42] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[43] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," 2022.

[44] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[45] M.-R. Amini, V. Feofanov, L. Pauletto, L. Hadjadj, E. Devijver, and Y. Maximov, "Self-training: A survey," 2022.

[46] S. Oymak and T. C. Gulcu, "Statistical and algorithmic insights for semi-supervised learning with self-training," *CoRR*, vol. abs/2006.11006, 2020.

[47] Y. Li, H. Li, C. Guan, and Z. Chin, "A self-training semi-supervised support vector machine algorithm and its applications in brain computer interface," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 1, pp. I–385–I–388, 2007.

[48] C. Wei, K. Shen, Y. Chen, and T. Ma, "Theoretical analysis of self-training with deep networks on unlabeled data," *CoRR*, vol. abs/2010.03622, 2020.

[49] T. Cheng, X. Wang, S. Chen, Q. Zhang, and W. Liu, "Boxteacher: Exploring high-quality pseudo labels for weakly supervised instance segmentation," 2023.

[50] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023.

[51] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr. 2024.

[52] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[53] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017.

[54] Riis, "Aerial sheep dataset." https://universe.roboflow.com/riis/aerial-sheep, jun 2022. visited on 2024-12-19.

[55] B. G. Weinstein, L. Garner, V. R. Saccomanno, A. Steinkraus, A. Ortega, K. Brush, G. Yenni, A. E. McKellar, R. Converse, C. D. Lippitt, A. Wegmann,

N. D. Holmes, A. J. Edney, T. Hart, M. J. Jessopp, R. H. Clarke, D. Marchowski, H. Senyondo, R. Dotson, E. P. White, P. Frederick, and S. M. Ernest, "A general deep learning model for bird detection in high resolution airborne imagery," august 2021.

[56] workspace1, "aerial-car-detection dataset." https://universe.roboflow.com/workspace1-3gwuj/aerial-car-detection-jca4t-jtccd-tqhsy, dec 2024. visited on 2024-12-19.

[57] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[58] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[59] C. Gomila and F. Meyer, "Graph-based object tracking," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 2, pp. II–41, 2003.

[60] F. Zeng, B. Dong, T. Wang, C. Chen, X. Zhang, and Y. Wei, "MOTR: end-to-end multiple-object tracking with transformer," *CoRR*, vol. abs/2105.03247, 2021.

[61] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," *CoRR*, vol. abs/2101.02702, 2021.

[62] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016.

[63] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, p. 1–10, 2008.