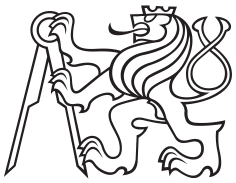
Bachelor Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Recognizing Unknown Objects for Open-Set 3D Object Detection

Nikita Sokovnin

Supervisor: doc. Ing. Tomáš Pajdla, PhD.
Field of study: Open Informatics
Subfield: Artificial Intelligence and Computer Science
May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Sokovnin Nikita** Personal ID number: **483744**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Recognizing Unknown Objects for Open-Set 3D Object Detection

Bachelor's thesis title in Czech:

Rozpoznávání neznámých objektů pro detekci 3D objektu ve světě bez omezení

Guidelines:

1) Study the approach to object detection and semantic instance segmentation [1] and to representing unknown object classes in open-set object detection setup [2].
2) Suggest an approach to detecting and representing objects unknown to the approach [1].
3) Implement the approach using simulation environment [2] and evaluate it on real data.

Bibliography / sources:

[1] Bolya, Daniel and Zhou, Chong and Xiao, Fanyi and Lee, Yong Jae. YOLACT: Real-Time Instance Segmentation of the IEEE/CVF International Conference on Computer Vision ICCV 2019.
[2] Dhamija, Akshay and Gunther, Manuel and Ventura, Jonathan and Boult, Terrance. The Overlooked Elephant of Object Detection: Open Set. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2020.
[3] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, Dhruv Batra. Habitat: A Platform for Embodied AI Research. ICCV 2019.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D., Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **08.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

doc. Ing. Tomáš Pajdla, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I am grateful to my supervisor doc. Ing. Tomáš Pajdla, Ph.D., who has offered an exciting project and guided me along the way. His interest in my work and study area motivated me to work hard and push my boundaries of knowledge.

I would also like to show my gratitude to my colleagues from Incognite lab, especially Mgr. Michal Vavrečka, Ph.D. and Ing. Megi Mejdrechová, for discussions and suggestions for numerous improvements.

I am thankful to my parents for their support and my roommate Aleksandr Shatrovskii for his helpful comments.

I gratefully acknowledge CTU for giving me a fundamental knowledge base for further development and CIIRC for allowing me to participate in research projects and providing computational resources.

This research was supported by the EU H2020 SPRING No. 871245 project and EU Structural and Investment Funds, Operational Programe Research, Development and Education under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000468).

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 21. May 2021

Abstract

While most studies in computer vision focus on closed set recognition, where the number of possible categories is fixed and known a priori, this study provides new insights into open set recognition. Open set problems assume that there is an unlimited number of categories, and most of them are unknown. This work contributes to our understanding of the unknown and how it can be applied in the object detection setting. To date, research on open set recognition has been focused chiefly on images or sequences of images. The novelty of our work lies in an adaption of 3D object detection to an open set setting. We analyse the detections from the popular instance segmentation framework, discuss the object detection performance on different examples, and show how unknown objects can be detected. We describe our approach to build a 3D open set object detection system, implement it in the simulation, and provide tools for evaluation. Moreover, this work demonstrates the method of learning novel classes without manual data labelling. We hope that the results of our work will bring robots closer to the ability to know that they do not know, and benefit from this finding.

Keywords: open set recognition, object detection, semantic segmentation, computer vision

Supervisor: doc. Ing. Tomáš Pajdla, PhD.

Abstrakt

Zatímco většina výzkumných prací v oblasti počítačového vidění je zaměřena na rozpoznávání v omezeném světě, v němž je počet možných kategorií pevně stanovený a předem daný, tato práce poskytuje nahlédnutí do rozpoznávání ve světě bez omezení. Úlohy ve světě bez omezení předpokládají, že počet kategorií je neomezený, a většina z nich je neznámá. Tato práce přispívá právě k porozumění neznámého a toho, jak může být použito pro detekci objektů. Výzkum rozpoznávání ve světě bez omezení je v současnosti zaměřen zejména na obrázky nebo na posloupnosti obrázků. Originalita této práce spočívá v adaptaci detekce 3D objektů do světa bez omezení. V práci jsou analyzovány detekce z populárního frameworku pro segmentaci instancí objektů, je diskutována kvalita detekcí objektů na různých příkladech a je dále ukázáno, jak lze detekovat neznáme objekty. V práci je navržen a popsán systém rozpoznávání 3D objektů ve světě bez omezení, jeho implementace v simulaci a prostředky pro vyhodnocení. Kromě toho tato práce demonstruje metodu učení nových tříd bez ručního anotování dat. Výsledky této práce mají potenciál přivést roboty blíž ke schopnosti vědět, že něco neví, a možnosti tento poznatek využít.

Klíčová slova: rozpoznávání ve světě bez omezení, detekce objektů, segmentace instancí objektů, počítačové vidění

Contents

Chapter 1

Introduction

1.1 Motivation

When researchers develop new technologies, they are usually inspired by natural human behaviour because it is easy to test some hypothesis by observing the decisions we are making in different life situations. Imagine that we walk around some object that we have never seen before, trying to determine what class it belongs to. Even for a human, sometimes one glance at the object is not enough to correctly classify it. 90% of the time while observing an object, we can name it confidently, but we cannot determine what it is just from a few viewpoints. Based on these observations, we will probably conclude that this object is, for example, a table and will use it as intended. However, what if, from each point of view, the object looks differently? In such a case, the logical conclusion would be that this object is unknown. This thought experiment has led us to believe that robots can make the same decisions by themselves in similar scenarios. They can exploit multiple observations made from different viewpoints. The capability of identifying unknown objects will allow robots to continuously learn new object classes only by interacting with humans and the environment. Introducing an "unknown" class to modern systems can prevent them from making wrong or dangerous decisions. If a robot sees a grenade for the first time, it is much safer for it to classify this object as "unknown" than give it the label "apple", even though some grenades look very similar to apples.

Recent research in computer vision is mainly aimed at closed set problems, where there is an assumption that the train and test split of the dataset

contain the same set of possible categories. Many modern recognition systems assume the world is static and closed, so all classes are known a priori. For this reason, most of the recent solutions are forced to choose between a small closed set of possible decisions. This often leads to unwanted consequences such as wrong decisions about novel objects. For example, many object detection systems cannot deal with an unlimited number of classes in the real world, as noted in [10].

There are works that reformulate the recognition setting to an open set [42, 1]. Open set problems assume that there are unknown categories that should be treated differently from known ones. Moreover, the identified unknown objects can be incorporated into the system to extend the knowledge about existing classes. Some works already made decent progress in open-set problems [42, 2, 31, 32], but most of them used academic datasets containing only unrelated images. By contrast, we extend an open set setting to a 3D world, where much more valuable features can be extracted. Unlike previous studies, we exploit the appearance of the same object in different conditions to reason about it.[1]

1.2 Our contribution

Our key contributions are:

- We study open set recognition from the new point of view. We show how outputs from the object detector can be used to determine whether some 3D object is unknown.
- We formalise Open Set 3D Object Detection. Based on previous works in object detection and open set recognition, we define the new problem and discuss possible solutions.
- We propose our approach and create simulation with close-to-real conditions for evaluation. We conduct experiments in simulation to demonstrate the performance of the proposed approach
- We suggest a method for learning novel classes and describe the dataset generation process without manual data labelling.

[1] The code is accessible at https://github.com/sokovninn/learning-unknown

1.3 Short overview of chapters

This work is organised as follows:

- In Chapter 2 we review and categorise recent advances in open set recognition, object detection and semantic segmentation to build a theoretical base for our approach.
- Chapter 3 discusses the representation of unknown objects, formalises our problem as open set 3D object detection and proposes our approach to solve this task.
- In Chapter 4 we describe semantic segmentation solution used to reason about single images, our modifications and talk about other required tools to build the whole pipeline.
- Chapter 5 shows the results of detecting 3D objects in an open set setting.
- Finally, we summarise our results and talk about future steps in open set 3D object detection in Chapter 6.

Chapter 2

Related work

2.1 Open set and open world recognition

If we make a common assumption that the labels are drawn from the same space during the train and test time, then we assume that the given environment is static. This refers to a closed set recognition. There is significant progress in solving such tasks with a finite number of classes. However, this assumption does not hold for the real world in which there is an unlimited number of labels that appear only during the test time. Adding an "unknown" class requires a classifier to classify the seen classes while dealing with unseen ones accurately. This type of task refers to open set recognition [13]. In real-world scenarios, the robot only has incomplete information about the world and needs to successfully divide objects into known and unknown to be able to acquire knowledge about previously unseen objects.

2.1.1 Open set classification

The main difference between open set and multiclass classification is that open set classifiers must label data far from any known class as unknown. The capability of rejection is very crucial in this setting. In [42] this problem is firstly formalised, and a 1-vs-set machine based on SVM is developed to deal with unknown classes. In this study, the open set risk was introduced to measure the risk of incorrectly labelling samples far from known classes. In [2]

OpenMax approach was introduced, which estimates the probability of input being unknown. OpenMax extends SoftMax to an open set using Weibull CDF probability on the distance between the final feature vector and vectors, representing all classes computed at the training time. In the following work [12] Generative OpenMax was proposed. This approach uses generative adversarial networks (GANs) to synthesise images from an unknown class.

In [27], it was pointed out that most of the existing datasets comprise so-called head, tail, and open classes. Head classes are the most represented classes. Tail classes are underrepresented, and open classes are all other classes that may be encountered at the test time. The distribution of these classes is often long-tailed and open-ended. Therefore, the nature of modern datasets can lead to confusion between the tail and open classes if we use a method such as OpenMax. The approach in [27] relates visual features between head and tail classes and more successfully rejects unknowns. In recent works, self-supervised [34] and unsupervised [48] approaches were studied to learn more informative features that help to distinguish known and unknown classes. Works [9, 21, 36] consider the open set classification problem as out-of-distribution detection, where the terms in-distribution (or inlier) and out-of-distribution (or outlier) examples are used to describe the open set nature of real data.

2.1.2 Open world classification

Open world setting requires a recognition system to be capable of continuously finding novel classes and being able to update itself with additional object categories with minimum downtime. Open world recognition was presented and formally defined in [1]. In this setting, the system should be able to classify known and unknown classes correctly and incrementally learn new classes using labelled instances, e.g. provided by a human. This type of scalable system can continuously update itself and increase its knowledge about the open world.

In [1] Non-Outlier (NNO) algorithm was presented. NNO extends existing incremental learning approaches that use Nearest Class Mean (NCM) [29, 39] by transforming the closed set problem to an open set. One representative example of an open world problem is described in [35], where the NCM principle was combined with object tracking and face recognition to incrementally identify new faces, memorise them, and then track these faces on video.

2.1.3 Open set detection

In [10] Dhamija *et al.* noticed that though many modern object detectors perform well on academic datasets, they fail in the real world. Most detectors produce false detections identifying unknown objects as some of the known classes with high confidence. Although these systems were trained to reject everything other than classes of interest, e.g. by using additional background class [14, 38, 26] that represents area without any object, unknown samples are usually classified as known ones. These types of errors are referred to as false positives or open set errors. Dhamija *et al.* [10] states that open set detection is just starting to develop and is far from being solved.

However, there are some attempts to solve this problem. One of the promising approaches is extracting epistemic uncertainty from the deep learning model. Epistemic uncertainty is caused by lack of knowledge and can be reduced by providing new data [8]. Some of the most relevant approaches are based on sampling-based techniques such as Monte-Carlo Dropout [11], or Deep Ensembles [20]. In [31, 30] MC Dropout is used, that is, performing inference multiple times while dropout is enabled and then evaluating multiple results. In [20] several distinct models process the data, and their outputs are combined to measure uncertainty. However, sampling-based techniques are computationally expensive as they process the same data multiple times and thus cannot be used in real applications. In [33] Miller *et al.* propose GMM-Det method that requires only one inference to measure uncertainty. This is achieved by adding an anchor loss [32], which is used for more appropriate clustering in the feature space, to the total loss during training.

2.1.4 Open world detection

The following work is the most relevant for our task. Open World Object Detection problem was recently formulated in [17]. In this setting, the model is tasked with detecting both known and unknown classes and incrementally learning novel classes without forgetting the old ones. This problem is the most challenging compared to the ones described above because it includes three independent research areas: object detection, open set recognition, and incremental learning. The solution in [17] is based on modifying the object detector by 1) adding an explicit unknown class to the dataset, 2) adding the contrastive clustering term to the loss, which helps to learn discriminative clusters, 3) adding energy-based unknown identifier, and 4) storing a balanced set of representatives of each class to mitigate forgetting.

2.2 Object detection

The task of object detection is to localise objects on the image by assigning to them tight bounding boxes and confidence scores. At the same time, locations with no objects should be ignored. There are two main types of object detectors: two-stage and one-stage. Fast R-CNN [14], and Faster R-CNN [38] are some of the representatives of the two-stage approach. The main idea is that firstly CNN processes the image to generate regions of interest (RoI). Then, fixed-length vectors are extracted from each RoI. And finally, a sequence of fully connected layers outputs the classification and bounding box for each RoI. This approach is sequential and, therefore, hardly parallelisable. Thus, in this case, it is hard to achieve real-time speeds. One-stage approaches like YOLO [37], SSD [26] or RetinaNet [23] focuses primarily on speed while sacrificing accuracy. One-stage approaches remove the second stage and leverage the accuracy gap in other ways, e.g. by strong data augmentation.

2.3 Instance segmentation

Computer vision tasks of this type are more challenging than others because they combine other two kinds of problems: 1) object detection, where the goal is to classify each object in the image and find the corresponding bounding boxes, and 2) semantic segmentation, where each pixel should be classified into a set of labels without differentiating object instances. Many of the existing methods to solve instance segmentation are two-stage, which means they are built from 2 sequential blocks. Firstly, they localise objects using bounding boxes, and then inside these boxes, semantic segmentation is applied.

Approaches like Mask R-CNN [15] can not achieve real-time performance. A better solution for real-time instance segmentation is YOLACT [3], which is used in our work. YOLACT is a one-stage method that is less accurate but is much faster than most state-of-the-art techniques. YOLACT does the same thing as SSD [26] or RetinaNet [23] do for the object detection - it removes the second stage and makes a higher level of parallelisation possible. Moreover, there are some extensions of YOLACT like YOLACT++ [4] or YolactEdge [25] that improve the model in both accuracy and speed. The general idea under the object detection and instance segmentation is the same - localisation of objects in the image. Although YOLACT is an instance segmentation model, we will refer to it as an object detector.

Chapter 3

Our approach

This chapter is dedicated to the formal definition of the problem and a general description of how our method works.

3.1 Formalising unknown

In the object detection setting, according to [43], the infinite set of possible object classes can be divided into three categories:

1. *Known classes*, i.e. classes that have labelled instances in the dataset (positive examples that the detector is trained to detect).
2. *Known unknown classes*, i.e. classes for which examples are present in the dataset and that are explicitly or implicitly labelled as negatives. Usually, they are part of the background, and the detector is trained to ignore them.
3. *Unknown unknown classes*, i.e. classes unseen at training time. The detector is not trained to ignore them.

In the open set object detection, there are the following types of errors:

- Closed-set errors

  Known class misclassification

  Duplicate classification

  Background (known unknown) detection

- Open-set errors

  Detection and misclassification of instances of unseen classes (unknown unknown)

The problem of this categorisation is that it is almost impossible to determine whether some unknown object is present in the dataset or not. Let us suppose we have a large-scale training dataset without labelled instances of the class "headphones". If there is at least one object "headphones" in one of the thousands of images, the class "headphones" automatically becomes known unknown. If we will detect this unknown object as known, it is hard to define the type of error: open-set or closed-set. To determine the type of error, we need to label each of the millions of objects in the dataset, which is a highly time-consuming process.

In [10] there is a slightly different definition of the unknown with respect to objects. The infinite space of labelled objects $\mathcal{Y}$ can be divided into:

1. *Known* $\mathcal{K} \subset \mathcal{Y}$

   $\mathcal{K}_K \subset \mathcal{K}$: *Known known*, i.e. object views present in the dataset. The model is trained to detect them.

   $\mathcal{K}_U \subset \mathcal{K}$: *Unknown known*, i.e. novel views of trained objects. They are usually contained in a test set.

2. *Unknown* $\mathcal{U} = \mathcal{Y} \setminus \mathcal{K}$

   $\mathcal{U}_K \subset \mathcal{U}$: *Known unknown*, i.e. background, garbage or undesirable object views. The detector is trained to ignore them.

   $\mathcal{U}_U = \mathcal{Y} \setminus (\mathcal{K}_K \cup \mathcal{K}_U \cup \mathcal{U}_K) = \mathcal{U} \setminus \mathcal{U}_K$: *Unknown unknown*, i.e. views of objects, which classes are not represented in the training dataset.

Modern object detection datasets contain many unlabelled objects, and it is hard to guarantee that object in the test is truly from $\mathcal{U}_U$. To address this problem Dhamija *et al.* [10] introduce an additional category **mixed unknown** $\mathcal{U}_M \subset \mathcal{U}_K \cup \mathcal{U}_U$. This category represents unlabeled and unseen object views that are not truly background (e.g. sky, grass, walls, or unseen

background). As mentioned above, the presence of mixed unknowns cannot be easily avoided, so modern solutions must be capable of dealing with them.

Before defining the unknown, we need to clarify what we understand under the term "class". All possible semantic classes can be divided into things and stuff [5]. Thing classes have a specific size, shape, and identifiable parts (e.g. car, person). Stuff classes are amorphous and do not have distinct parts (e.g. sky, grass). Our work aims to identify things, so in the following text, by "class" we mean "thing class".

We use the strong sides of both approaches and define the unknown as follows:

1. **Known classes** K: classes that have labelled instances in the training dataset. A known object (or simply known) is an instance of a known class.
2. **Unknown classes** U: classes which are not trained, no example of these classes is labelled. Instances of these classes might be present in the background at training time. U is very similar to mixed unknown $\mathcal{U}_M$. An unknown object (or simply unknown) is an instance of unknown class.

3.2 Problem specification

The closest problem setting to ours is Open World Object Detection introduced in [17]. Let us formalise our task definition in this section.

3.2.1 General problem

First of all, we have a set of known classes $K = \{1, \ldots, C\}$ where C is the number of classes present during the training time. Furthermore, there is an unlimited set of unknown classes $U = \{C+1, \ldots\}$ which may be encountered during the inference. Let 0 be the label associated with an "unknown" class. Each known class is by assumption represented in the dataset $D = \{X, Y\}$ with cardinality $|D| = M$ where X is a sequence of images $X = \{I_1, \ldots, I_M\}$ and Y is a sequence of matching labels $Y = \{L_1, \ldots, L_M\}$. L_i encodes k_i

instances $y_1, \ldots, y_k$ present in an image I_i. Each label y_j is represented as $y_j = [u, v, w, h, c, s]$ where u, v are the coordinates of the center of bounding box, w, h are the width and height of the bounding box, c is the class of the object and s is a segmentation mask (can be omitted for default object detectors). The above refers to the object detection model M_d, which produces predictions for a single image in each timestep. Thus, it has neither the capability of reasoning about image sequences nor comprehension of the 3D environment.

The observer is placed into the real or simulated 3D scene, which contains N_o objects. Each object is represented by a pair (d, l) where d is a 3D coordinate of the object position and l is a label from the set $S = \{0\} \cup K$. The choice of the object representation will be discussed further in Section 3.5.2. The observer can move around the scene to observe objects and reason about them. In each timestep t, the observer can receive the information from its sensors and process it.

The task is to localise and correctly assign a label from S to each object in the scene by a metaclassifier M_m.

The problem specification assumes that the observer can access both the object detector M_d and a meta-classifier M_m. The meta-classifier can use all the information observed from the scene and all outputs of the object detector. In essence, M_m is intended to solve the core task, which is: based on M_d outputs made from the different viewpoints for a single object $\{y'_1, \ldots, y'_{N'}\}$, decide whether an object is known, formally has a class from the set K, or has a class "unknown".

When the observer finds unknown objects $\{u_1, \ldots, u_n\}$ it can ask the oracle to provide true labels of these objects and optionally query the dataset with training examples containing only instances of the found unknown objects to update M_d. Alternatively, the observer can construct a dataset by itself, for example, by doing a 3D reconstruction of the objects and creating a dataset using simulations with a wide range of augmentations. Retraining should extend the observer knowledge about existing classes. Formally, after updating itself with new classes, M_d can identify test instances belonging to some class from the actualised set $K' = K \cup \{C + 1, \ldots, C + n\}$. Therefore, M_m can assign some of the classes $S' = \{0\} \cup K'$

3.2.2 Meta-classifier formalisation

Formally meta-classifier M_m should be measurable classification function $f : \mathbb{R}^d \mapsto S$, where $f \in \mathcal{F}$ and d is the dimension of the feature vector extracted from multiple outputs of M_d. $\mathcal{F} : \mathbb{R}^d \mapsto S$ is suitably smooth space of classification functions. Let $x \in \mathbb{R}^d$, $y \in K$, $f_y(x) > 0$ when class y is recognised, $f_y(x) = 0$ when class y is not recognised (classified as unknown). The core of our open-set recognition task is to find a function f that minimises the open set risk defined in [42]. Let us adopt the general definition of open set risk.

One part of the task is to not label unknowns as knowns. For this, we can define the open space risk. If we consider the space of all features S_O as a large ball, which contains both open space O of unknowns and all of the positive training examples (knowns), open space risk $R_O(f)$ for class y can be defined as follows:

$$R_O(f_y) = \frac{\int_O f_y(x)\ \mathrm{d}x}{\int_{S_O} f_y(x)\ \mathrm{d}x}$$

According to [43] open space O is the space sufficiently far from all known positives examples $[x_1, \ldots, x_N]$ and can be defined as follows:

$$O = S_O - \bigcup_{i \in N} B_r(x_i)$$

where $B_r(x_i)$ is a closed ball of radius r centered around training example x_i.

The more we label open space as known, the greater is the open space risk. Open space risk is considered to be the relative measure of positively labeled open space compared to the overall measure of positively labeled space (which includes the space near the positive examples). We need to minimise it. At the same time, we want to minimise the empirical risk R_ϵ on the training set T to keep the prediction accurate in closed set terms. T might contain both labelled knowns and unknowns or only knowns.

Our goal is to balance the open space (unknown) risk with the empirical

(known) risk. In other words, the optimal function f can be found by minimising the open set risk:

$$f = \underset{f}{\operatorname{argmin}}\{R_O(f) + \lambda R_\epsilon(f(T))\}, \text{where } \lambda \text{ is normalisation constant}$$

3.3 Unknown classification discussion

We consider unknowns as objects that belong to classes that are not sufficiently represented by M_d. We exploit the false positive errors of the detector to reason about the objects (Figure 3.1). There are 3 main reasons that might cause problems with class representation:

- Absence of labelled instances.
- Small diversity of possible object appearances in the dataset.
- Weak generalisation capacity of the detector.

In addition, some class representations in the feature space may be very close to each other, e.g. “skateboard” and “surfboard”. In this case, we can measure the similarity between these classes and make the decision based on it. If we train the classifier for this task, we should consider class similarity when computing the loss. However, it is out of the scope of our work.

3.4 Approach overview

The main components of our approach are illustrated in Figure 3.2. The core idea is that at each timestep, YOLACT processes an image and outputs detections. These detections are then backprojected to 3D and aggregated into clusters. When clusters contain enough detections, the meta-classifier makes the decision based on all object observations. The final result is the position of the object and the assigned label from the set $S = \{0\} \cup K$.

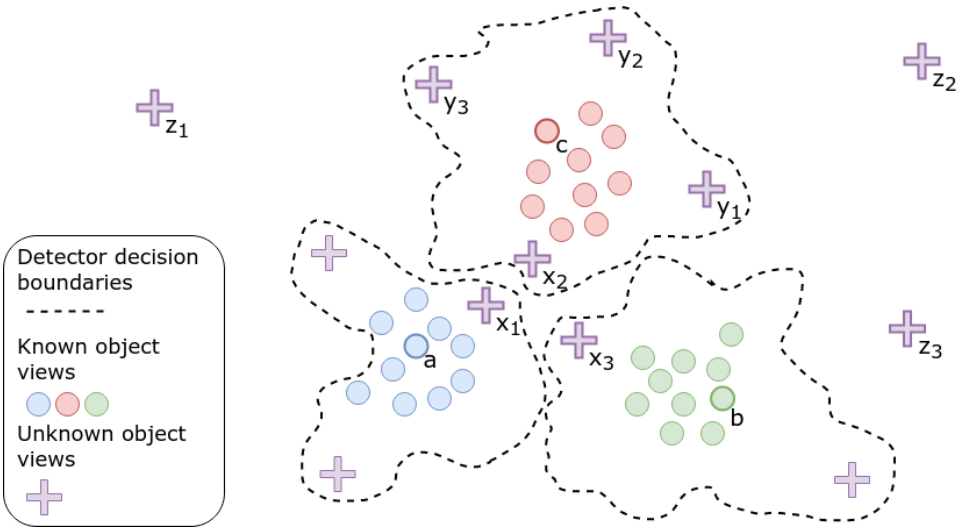


Figure 3.1: Open set problem illustration. Each point represents a different view of some object. The closer a point is to the centroid of the blob - the more confidence score is. a, b, c are views of known objects classified correctly almost all the time. Unknowns can be divided into several groups by their distance to decision boundaries and their properties. x_1, x_2, x_3 - views of an unknown object that often has confusing labels and does not have a clear classification. y_1, y_2, y_3 - views of an unknown object that has a consistent classification in different conditions but with low confidence. z_1, z_2, z_3 - views of the unknown object that are far from all known classes and most of the time "invisible" for the detector. Such unknowns are undetectable, which means that they are consistently classified as background.

3.5 Approach details

In this section, we discuss each part of the system in detail.

3.5.1 Backprojection from images to 3D

Our system needs to have access to the absolute 3D coordinates of the points captured by RGB sensors. Understanding the nature of 3D objects and the ability to measure the distance between them is essential in our task. When an RGB camera captures an image of the scene, the agent loses depth information. This happens because 3D points are mapped onto a 2D plane. Depth information, which is essential in our case, can be obtained using stereo cameras or depth sensor. In the simulation, our agent has access to a depth image of the scene. Given the RGB image, depth image, and sensor positions, we want to make an inverse projection from the 2D to 3D space. It can be

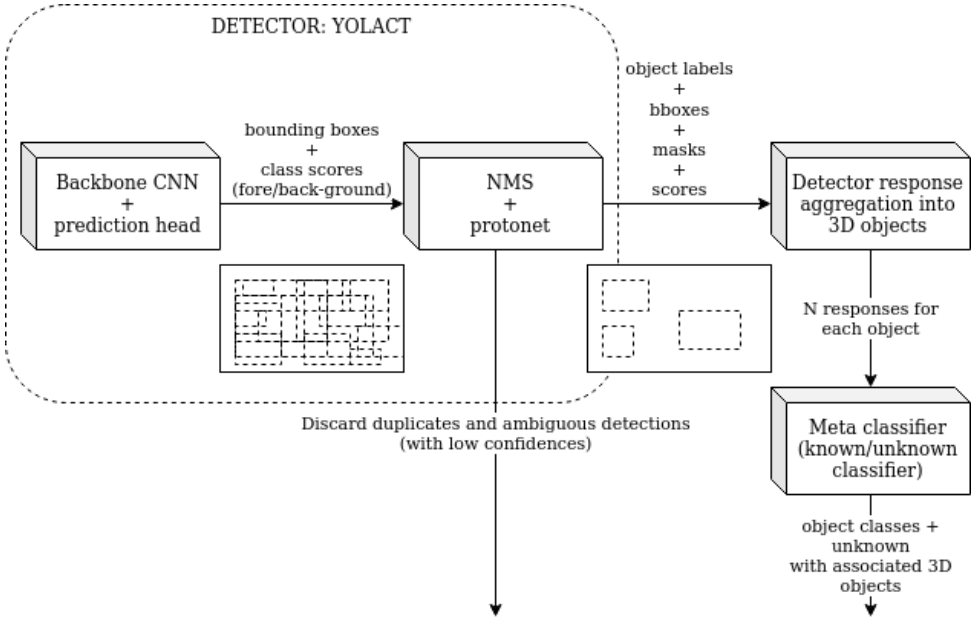


Figure 3.2: Scheme of our approach. Details of the YOLACT architecture will be discussed in Section 4.1.1.

easily done by using inverse extrinsic and intrinsic matrices of cameras.

Firstly, we need an intrinsic camera matrix K that describes the camera's internal parameters:

$$K = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y represent focal length, p_x and p_y represent principal point, s is the skew coefficient between x and y axis.

In further computations, we need our intrinsic matrix to have a shape 4x4, so we add an extra row $[0, 0, 0, 1]$. Moreover, we flip the sign of the element at position $(3, 3)$ to correspond to the coordinate system in the simulation. With the simulated camera parameters, our final intrinsic matrix is:

$$K' = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To compute an extrinsic matrix, we need a rotation matrix R that describes the camera's orientation with respect to the world coordinate system and

column vector T describing the camera's centre in world coordinates:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Now we can produce an extrinsic matrix:

$$\begin{bmatrix} R^T & C \\ \mathbf{0} & 1 \end{bmatrix}$$

where $C = -R^T T$ and extra row $[0, 0, 0, 1]$ is added to make the matrix square.

Finally, to compute the 3D coordinates $[x_w, y_w, z_w]$ of the 2D point on RGB image with coordinates $[x, y]$ in pixel space and its depth on depth image z in meters, we use this equation:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} R^T & C \\ \mathbf{0} & 1 \end{bmatrix}^{-1} K^{-1} \begin{bmatrix} x\,z \\ y\,z \\ z \\ 1 \end{bmatrix}$$

3.5.2 Object representation and clustering

In the standard formulation of 3D object detection, the task is to assign a 3D bounding box to localise the object. Our approach is much simpler. We use the centroid of the first mask given to the object in the image as an initial centre of the cluster. The cluster is a sphere that represents the object. The 3D absolute coordinate of the centroid is then associated with the predicted label. All subsequent detections with centroids close enough to some of the existing cluster centres are assigned to this cluster. New detections assigned to the existing cluster update the centre incrementally as a mean of all assigned mask centroids.

However, this representation simplification may lead to unwanted consequences. If the object is too big, it may have many labels for different parts because the centroids received from this object are too far from each other. On the other hand, small objects, which are close to each other, can merge. A better way to represent objects is to store point clouds or 3D bounding boxes, but these approaches are harder to implement. For example, point

clouds require a decision for every point in the mask received from YOLACT, to which object it does belong. These methods will be investigated in further research.

The algorithm of detection clustering is as follows:

```
Result: Clusters with assigned labels
cnn = init_yolact();
env = init_env();
clusters = {};
episode_over = False;
while not episode_over do
    image = env.step();
    detecions = cnn(image);
    centroids = compute_centroids(detections);
    assigned = False;
    for centroid, detection in zip(centroids, detections) do
        world_position = backprojection(centroid);
        cluster = find_corresponding_cluster(world_position,
         clusters);
        if cluster is None then
            cluster = init_cluster(world_position);
            clusters.append(cluster);
        else
            cluster.update_center(world_position)
        end
        cluster.append(detection);
    end
end
```

Algorithm 1: Clustering method pseudocode

3.5.3 Decision making

We use 2 different known/unknown classification approaches: confidence thresholding over the constructed feature vector and entropy measurement over the mean softmax vector.

Confidence thresholding

When we have enough detection outcomes from the detector assigned to one cluster, we can decide whether the object is known or unknown. We simulate the confidence with a normalised feature vector. If the confidence level of the most confident class known to the detector is smaller than a threshold, then we classify the object as unknown.

Formally, given:

- C labels $K = [1, \ldots, C]$ of object classes known to detector
- N classification outcomes (l_j, c_j) of an object classification by detector with label l_j and its confidence c_j
- a - C-dimensional vector of the sums of outcome confidences for each label K_k, i.e. $a_k = \sum_{\{(l_j,c_j) \text{ s.t. } l_j=k\}} c_j$

We compute the feature vector using the following:

$$v = \frac{[a_1, \ldots, a_C]^T}{\sum_{i=1}^{C} a_i}$$

The final classification rule is:

$$L^*(v) = \begin{cases} K_{argmax(v)}, & \text{if } max(v) > \theta_1 \\ unknown, & otherwise \end{cases}$$

where θ_1 is threshold from range $[0, 1]$

In the following text, we will refer to this method as “maxsum”.

Entropy as an uncertainty measure

Another possibility to reject unknown objects is to measure the epistemic uncertainty of the detector. Instead of using the top class’s final detection and

probability score, we can utilise the full score vector with all class probabilities. In [31] classification uncertainty is extracted from the final softmax vector q computed as mean of the n softmax scores $\{s_1, \ldots, s_n\}$.

$$q = \frac{1}{n} \sum_{i=1}^{n} s_i$$

In [31] q is treated as an average vector of class probabilities over a set of score vectors, produced by multiple forward passes of the same data with enabled dropout. In other words, it is the average result of the model ensemble. In our case, q represents the mean score vector over multiple views of the same 3D object. Therefore, uncertainty may arise when the detector is not able to recognise the object in different conditions identically. We work with a detector that produces a probability for a negative "background" class, so the dimension of q is $C + 1$, where C is the number of object classes. Let us set the "background" label as 0.

One of the ways to measure classification uncertainty is to use normalised entropy:

$$H(q) = -\sum_{i=0}^{C} \frac{q_i * log(q_i)}{log(C + 1)}$$

If q has a uniform distribution, the entropy will be maximal, so the uncertainty is high. If the class probability is concentrated in one class, the entropy will be low, which means that the detector is confident in its detection.

The classification rule in this case is:

$$L^*(q) = \begin{cases} (\{0\} \cup K)_{argmax(q)}, & \text{if } H(q) < \theta_2 \text{ and } argmax(q) \neq 0 \\ unknown, & otherwise \end{cases}$$

where θ_2 is threshold from range $[0, 1]$

3.5.4 Learning new objects and classes

One of the essential abilities of intelligent systems that function in the real world is learning. With the ability to identify unknown objects, such a system can query a human to provide a training set with an unknown or unknown class or create such a dataset. Before we explain our learning approach, we need to discuss which properties the learning process in such systems should have.

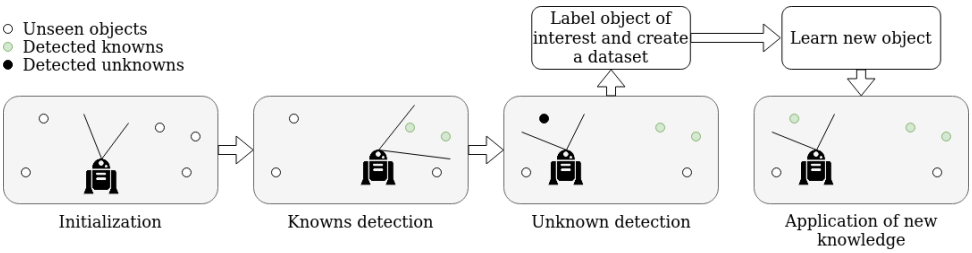


Figure 3.3: Scheme of learning.

Ideally, the learning process should be:

- Active: An active learner can query the oracle to label unlabelled data and use it to improve itself.
- Scalable: Not only knowledge about existing classes can be increased, but also new classes can be learned.
- Online: Learner can update itself in each step with data that become available in sequential order.

We use the option of constructing a dataset by a human. However, this process can be easily automated while it does not require manual labelling. Our solution is active, scalable, and offline. When an unknown object is found, we construct the dataset, append it to the training dataset and train the detector from scratch on the extended dataset (Figure 3.3). We do not annotate the new dataset manually. Instead, we use advanced tools to create it automatically. Thus, the whole learning process can be merged into one fully automated pipeline.

As discussed in Section 3.3 the first reason for assigning a label unknown to the object is an insufficient amount of training data in the dataset. In this case, we can finetune the detector to new instances of the existing class. In our experiments, we only show how the extension to an additional class works.

Chapter 4

Implementation

4.1 Toolkit

In this section, we describe the tools that we use to implement our solution and explain the choice of each one.

4.1.1 YOLACT

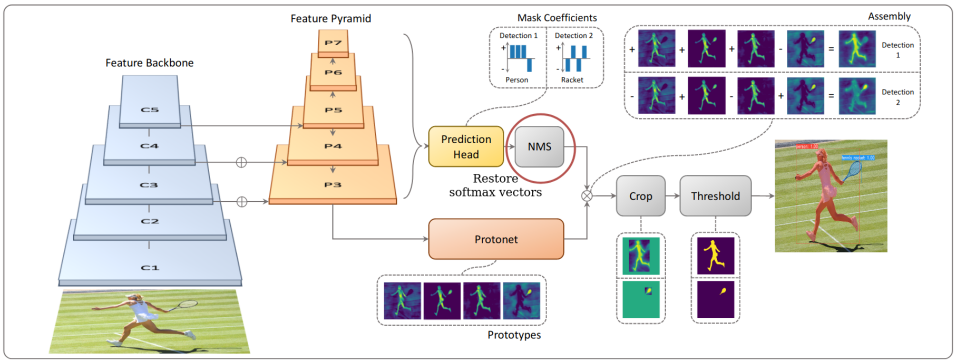


Figure 4.1: YOLACT architecture from [3]. Blue/yellow indicates low/high values in the prototypes. Grey nodes indicate functions that are not trained. We modify YOLACT in the Non-maximum Suppression part, where initially vectors with score probabilities are discarded, and only final detections are propagated further.

Since we want our system to work in real-life scenarios, object detection should be as fast as possible. YOLACT is a real-time instance segmentation framework that breaks up the main task into two parallel parts: generating prototype masks and predicting per-instance mask coefficients. Final instance masks are then produced as a simple linear combination of prototype masks with predicted coefficients (see Figure 4.2). A convolutional model trained in this way can achieve 29.8 mAP on MS COCO at 33.5 fps.

Figure 4.2: Visualisation of the YOLACT output

YOLACT does a sequence of decisions (Figure 4.1). First, it selects a subset of all image rectangles. These selected rectangles are called anchors or prior bounding boxes. Anchors have different scales and different ratios. They are distributed over the whole image and should represent the most frequent object positions on the image. Then for each prior bounding box, YOLACT produces shifts of the shape, prototype mask coefficients and class probability vector for $C + 1$ classes where C stands for the number of classes in the training set and 1 for the negative class. In fact, YOLACT classifies each anchor into one of the following classes:

- The "background" class which models all examples generated by hard negative mining. There are no explicit annotations of the background class in the training set. However, some of the anchors that do not contain annotated instances are selected to be representatives of the negative "background" class during training. The "background" class

is very broad and includes many samples that are often close to object class boundaries.

- The "object" classes that are obtained by manual labels from the COCO dataset. Object classes are more compact than the background class as they correspond to well-defined objects with coherent appearance.

Then most of the predictions that have too big overlap or too low confidence are filtered by Non-maximum Suppression without taking into account "background" class. The reason there is no "background" label in the final detection is that it displaces "object" classes and mostly does not contain any useful information (Figure 4.3). However, the "background" element of the probability vector can be used for detecting unknowns. Thus, we modify YOLACT to extract these probability vectors.

Figure 4.3: Background labels.

Next, we briefly describe the main parts of the YOLACT architecture depicted in Figure 4.1:

Anchors

Anchors (or default bounding boxes) were introduced in RPN [38] to address the problem of object detection at multiple scales. The idea is to predict

multiple region proposals with predefined aspect ratios and scales for each location of the sliding window on the final convolutional feature map. The right choice of the scale, aspect ratio and the number of anchors can improve the model performance, as they will better represent the most frequent object positions on the image. The number of anchors in YOLACT is approximately 20k.

Feature backbone

As a backbone feature extractor, YOLACT uses ResNet-101 [16] with Feature Pyramid Network (FPN) [22]. FPN is used to effectively generate a multiscale feature pyramid which helps to detect objects with different scales. A feature pyramid is a set of feature maps at different scales. Default bounding boxes of different aspect ratios are placed on three selected layers of FPN. Finally, we have features for default bounding boxes with different scales and ratios as an output of the backbone feature extractor.

Prediction head

The prediction head is attached to the three selected feature maps (to each default bounding box). YOLACT modifies the prediction head architecture suggested in [23] by adding a third branch for mask coefficients and making some of the convolutional layers shared. The prediction head in YOLACT is CNN, which has three branches to predict in parallel $C + 1$ class confidences, four bounding box regressors and k prototype mask coefficients. Bounding box regressors are shape offsets that are needed for anchor transformation and better localisation of the object.

Non-maximum Suppression

To suppress duplicate detections, YOLACT uses a modified version of NMS, which is faster but may filter more bounding boxes than needed. As we mentioned above, NMS in YOLACT considers all classes except the negative class “background” and therefore outputs only detections of positive classes. We modify NMS so that with bounding boxes and class confidence scores, it also returns the corresponding softmax vectors.

Protonet

Protonet is the CNN attached to the deepest feature map in FPN. Protonet predicts k prototype masks for the entire image. Each prototype mask activates on a different feature. Some localise instances, some detect instance contours, and some encode position-sensitive directional maps.

Mask assembly

Instance masks are computed as a linear combination of k prototype masks with k coefficients from the prediction head:

$$M = \sigma(PC^T)$$

where P is an h×w×k matrix of prototype masks and C is an n×k matrix of mask coefficients for n instances surviving NMS and score thresholding. Then the instance masks are cropped with the predicted bounding box at the evaluation time and the ground truth bounding box at the test time.

Hard negative mining

In the datasets of both instance segmentation and object detection tasks, there is usually a large imbalance between the number of annotated objects and the number of background examples (regions that are not annotated). This imbalance may cause low accuracy or slow training. One technique that can solve this problem is dataset bootstrapping [41] or typically called hard negative mining. The idea is to construct training data containing positive and "hard negative" examples. "Hard negative" instances are false positive detections of the negative class, which have high confidence scores and affect the loss most. To filter the excessive number of negative examples, YOLACT uses hard negative mining. When the classification loss is computed, only part of the negative examples (which contribute to loss the most) is taken into account. The ratio is 3:1 (negative:positive), which means that for one positive example, there are three negatives.

Understanding this approach is essential for us. If there is an unlabeled object in the image, it might be chosen as hard negative but might not. Thus, we cannot be sure whether the detector is trained to reject this object or not. The background region chosen for training is very sparse, so we cannot expect the detector to reject all unlabeled instances successfully. This may cause false positive errors on both types of unknown classes: those had unlabeled examples in the dataset and those that did not. We can use these false positives to detect unknown objects in 3D. Moreover, when we receive detection with a high object class, we consider background class as unknown since it can never faithfully represent all non-objects.

4.1.2 AI Habitat

The most straightforward way to teach an embodied AI agent is to train it in the real world. This method allows us to see the robot performance in real scenarios immediately. One of the most significant benefits of such a method is that the sim2real step is unnecessary. There is no need for additional modifications of the trained model. Moreover, the confidence that the agent will perform similarly in a test time as in a training time is high. However, this approach has many disadvantages. Some of them are:

- Experiments are relatively slow since we have no control over the flow of time.
- Some of the trials could be dangerous and destructive both for the robot and the environment.
- Many experiments are hardly reproducible due to the stochastic nature of the real world.
- Additional measurements might need additional sensors, which can be quite expensive.

3D simulators can help to deal with these problems. Platforms for training embodied agents based on 3D simulators have made significant progress in recent years. The reasons why do people use them in research are:

1. **High speed.** Processes in simulation can be parallelized so a larger amount of experiments can be conducted in the same interval of time compared to the real conditions.

2. **Easy to reproduce.** We can control all aspects of the environment and receive the same results each time while keeping the parameters unchanged.
3. **Easy to extend.** In many simulators, the processes of adding new robots, objects or scenes are effortless.
4. **Cheap.** The only thing we need to train embodied AI agent is a powerful enough machine to run the simulation.
5. **Safe.** Entire process runs inside the simulation and can be easily reverted.

In our problem setting, we assume that the environment is static, so no advanced physics simulation is required. As we work mainly on visual perception, fast and high-quality rendering is essential. For simulating robot movement in a photorealistic environment, we use the Habitat platform [28], which includes Habitat Sim and Habitat Lab. Habitat Sim is a high-performance 3D simulator with configurable agents, multiple sensors, and generic 3D dataset handling. Its main advantage is extremely fast rendering, which can achieve 10000 fps on a single GPU. Habitat Lab is a modular high-level Python library for developing AI in such tasks as robot navigation, instruction following, and question answering. After training and evaluating the agent in the simulation, we can transfer the acquired knowledge and skills to the real robot.

4.1.3 Microsoft COCO Dataset

One of the primary capabilities of our system is an understanding of visual scenes. Closed set problems are much more straightforward than open set because they do not require the model to reason about unknown object categories correctly. Nevertheless, models which solve such tasks still require a considerable amount of data. Qualitative and massive datasets are the main force power in advancing computer vision problems. In our work, the crucial role of solving semantic segmentation plays the Microsoft COCO dataset [24].

Microsoft COCO dataset is large-scale object detection, segmentation, and captioning dataset (see Figure 4.4). Some of the main features that lead many researchers to use this dataset for training and evaluation are:

- Object segmentation: objects are labelled using per-instance pixel-level segmentation.

Figure 4.4: Samples from the Microsoft COCO Dataset.

- Size: the dataset contains more than 200 thousands of labelled images and around 1.5 million object instances.
- Categories: 80 object categories are selected by their frequency of occurrence, usefulness in real applications, and diversity relative to other categories.
- Recognition in context: images are captured from noncanonical viewpoints and contain objects in their natural context.

YOLACT was trained on *train2017* and evaluated on *val2017* and *test-dev* splits of the COCO dataset.

4.1.4 RealityCapture

To transfer real objects to the simulation, we need software for producing high-quality 3D meshes. RealityCapture[1] solves this problem by offering a high-speed and easy to use method for scanning real objects and scenes. It offers tools for creating virtual reality scenes and textured 3D meshes. The process of object reconstruction from the images is almost fully automated.

[1] https://www.capturingreality.com

Additionally, we use Blender[2] to edit the final meshes produced by RealityCapture.

4.1.5 myGym

Training of YOLACT to recognise previously unknown objects requires additional data. To produce the dataset with annotated images for semantic segmentation, we use myGym [46] - the modular framework for developing and benchmarking RL algorithms. The main feature of myGym is the integrated visual module. This module allows working with tasks that incorporate visual recognition. Moreover, it provides an option to generate synthetic datasets for integrated computer vision models: YOLACT and VAE [19].

Dataset generator uses PyBullet [7] physics engine to construct scenes and synthesise images with objects in simulated environments. The generator is highly configurable and easy to use - camera settings, robots, object appearance, and other parameters can be controlled using a single configuration file.

Synthetic datasets generated in myGym are helpful not only for simulated scenarios but, more importantly, for real ones. Sim2Real transfer is possible due to the high level of data augmentation. A fast and straightforward technique that helps achieve sufficient accuracy on real photos is domain randomisation [44]. The primary purpose of this approach is to provide enough variability of simulated data at the training time such that the model will be able to generalise real-world data during the test time. The simulated environment can be randomised at several levels: random colours/textures applied to all objects, random camera position and orientation, variable light position, type, and intensity.

4.2 Implementation details

In this section, we discuss some of the implementation aspects.

[2] https://www.blender.org

4.2.1 Scanning of real objects

To reconstruct a 3D object from the images, we take 60 photos on average for each of the 10 chosen test objects. The choice of the objects was motivated by the presence of the class in the COCO dataset, the shape of the object, and the quality of reconstruction. For many objects that were initially in this set, good reconstruction was impossible for a few reasons: low number of visual features, transparency, and glossy surface. Figure 4.5 shows the scanning process.

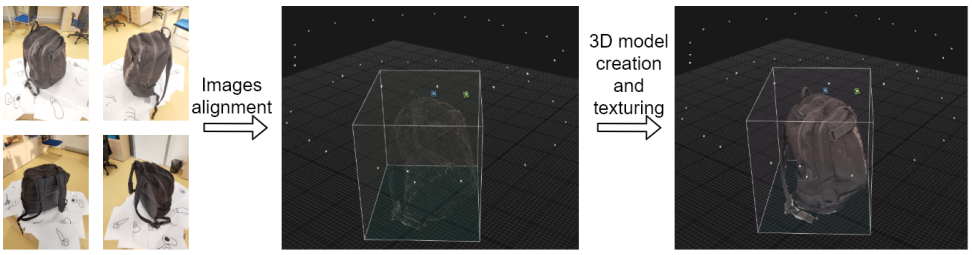


Figure 4.5: The process of scanning 3D object in RealityCapture.

4.2.2 Agent and objects in the simulated environment

To make our experiments fast and reproducible, we create a virtual copy of a room with an agent within it using a Habitat simulator. For the experiment, the indoor space was taken from Gibson dataset [47]. Inside the room, we place the scanned objects. Half of them are known objects, and another half is unknown (Figure 4.8). The agent is equipped with a set of sensors: 1 RGB sensor, 1 depth sensor (Fig. 4.6) position/rotation sensor. Agent can be controlled by a set of actions: “move forward”, “turn left”, “turn right”, “look up”, “look down”.

Figure 4.6: RGB and depth sensors inside Habitat simulation.

4.2.3 Dataset generation for an unknown object

With a mesh produced by RealityCapture software, we can create a new dataset using myGym (Figure 4.7). The size of the new dataset with a single class "dino" is 10800 images for train split and 1200 images for validation split. The dataset creation process in each of the 4000 iterations is as follows:

1. Reset simulation with random light conditions and textures
2. Initialise 3 cameras with RGB/semantic/depth sensors in specified positions with random shifts
3. Spawn an object in a random position and with random orientation
4. Render images from RGB sensors and corresponding semantic sensors
5. Save produced RGB images, semantic masks and bounding boxes in COCO format

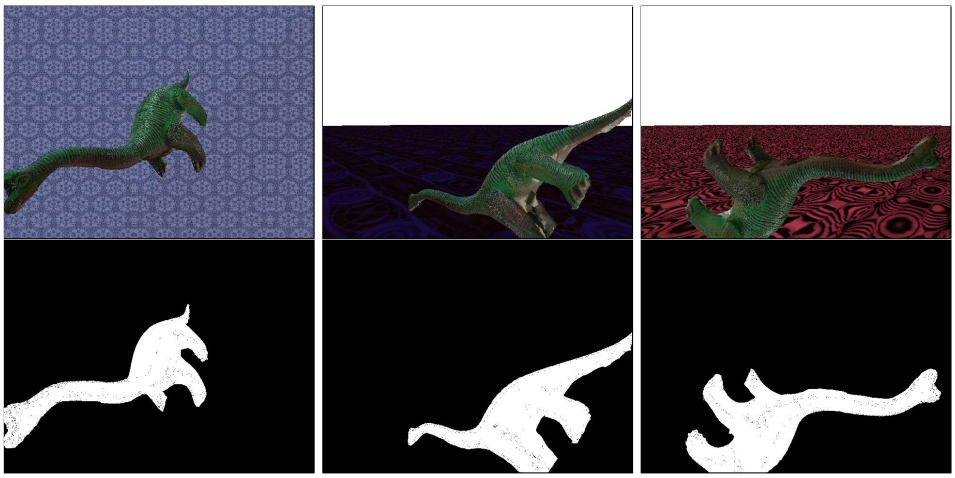


Figure 4.7: Samples from the generated dataset with corresponding semantic masks.

Figure 4.8: Object set transferred to Habitat simulator. Known are in the first row, unknown are in the second row.

Chapter 5

Experiments

In this chapter, we describe the conducted experiments and discuss the obtained results.

5.1 Evaluation metrics

Because our task is not widespread in computer vision, we need to describe all the metrics we use carefully. 4 essential components are used for metrics computation. These components are closely related to a classical object detection task:

- True Positive (TP) - this is a correct positive detection, that is if the label and prediction for the object at a certain position are the same.
- False Positive (FP) - wrong positive detection. This corresponds to the case when the object is misclassified, or there is no object at the predicted position at all.
- False Negative (FN) - this is the case when the object is not detected.
- True Negative (TN) - this implies an absence of prediction on the negative instance, i.e. the absence of any object. This metric does not apply to our task because there is an infinite number of instances that should not be treated as objects.

Because we can not obtain the True Negative metric, we are not able to work with the False Positive Rate, which is the required component for the receiver operating characteristic (ROC) curve.

Extension of micro F1 score to open-set

We use the following equation to compute a micro F1 score:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Open set modification of the F1 score was introduced in [18]. The most effortless adaptation of this formula to an open set scenario is to consider the unknown class to be just an additional class. However, this approach is not appropriate in open set tasks because we can not correctly treat classified unknown samples as True Positives. This does not make sense because there are no representative samples of the unknown class during training. That is why a much better choice in this case will be the modified precision and recall that can be computed in such way:

$$precision = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)}$$

$$recall = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FN_i)}$$

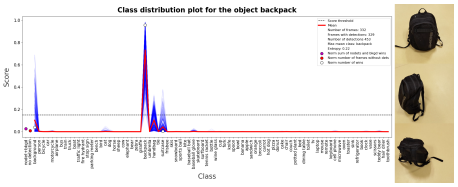
where n is the number of classes on which the model was trained (not including an "unknown" class).

We use the F1 score because it is invariant to True Negatives, which are not present in our task.

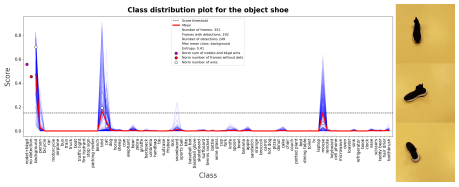
5.2 Score distribution study

To study how the data we work with look like, and get some insights, we visualise the distributions of probability scores for selected real objects. For each object, we created a video by walking in a circle around the object. All videos contain on average 360 frames at a speed of 30fps. For simplifying the process of analysing detections, each video contains only one object and a

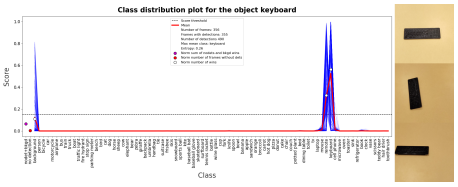
neutral background, which is not detected in most cases. YOLACT process each frame in the video and outputs detections, which we store for further analysis. We set the YOLACT confidence threshold to 0.15, which means that detections with a maximum class confidence value lower than 0.15 are discarded. Graphs for 10 test objects are in Figure 5.1, 0th, 100th and 200th frames from the corresponding video are on the right of each graph. Let us discuss the results for both known and unknown sets so that easy cases are considered first, and the most difficult ones are considered in the end.



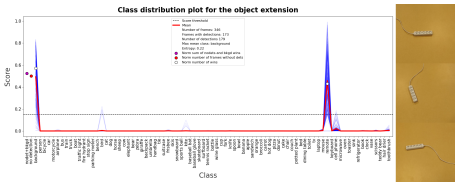
(a) : Strong “backpack” class.



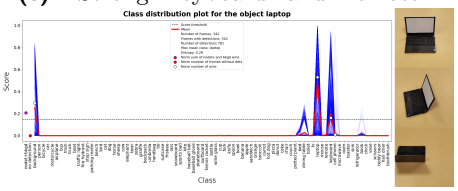
(b) : Strong “background” and weak “bird”.



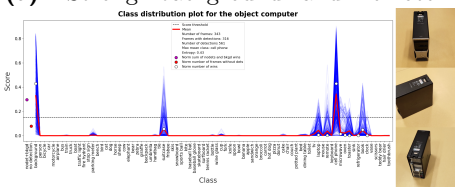
(c) : Strong “keyboard” and “remote”.



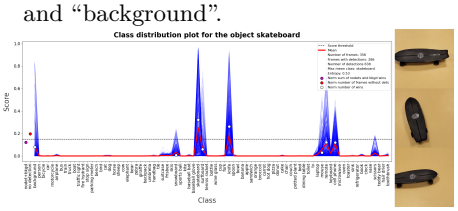
(d) : Strong “background” and “remote”.



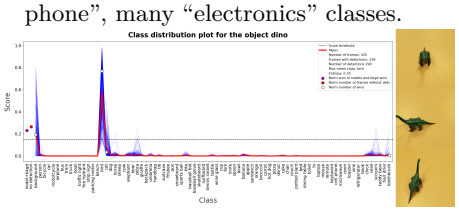
(e) : Strong “laptop”, weak “keyboard” and “background”.



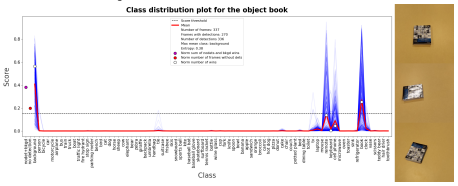
(f) : Strong “background” and “cell phone”, many “electronics” classes.



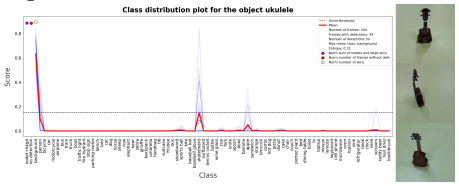
(g) : Weak “skateboard” and “knife” and many other classes.



(h) : Strong “bird” and weak “background” classes.



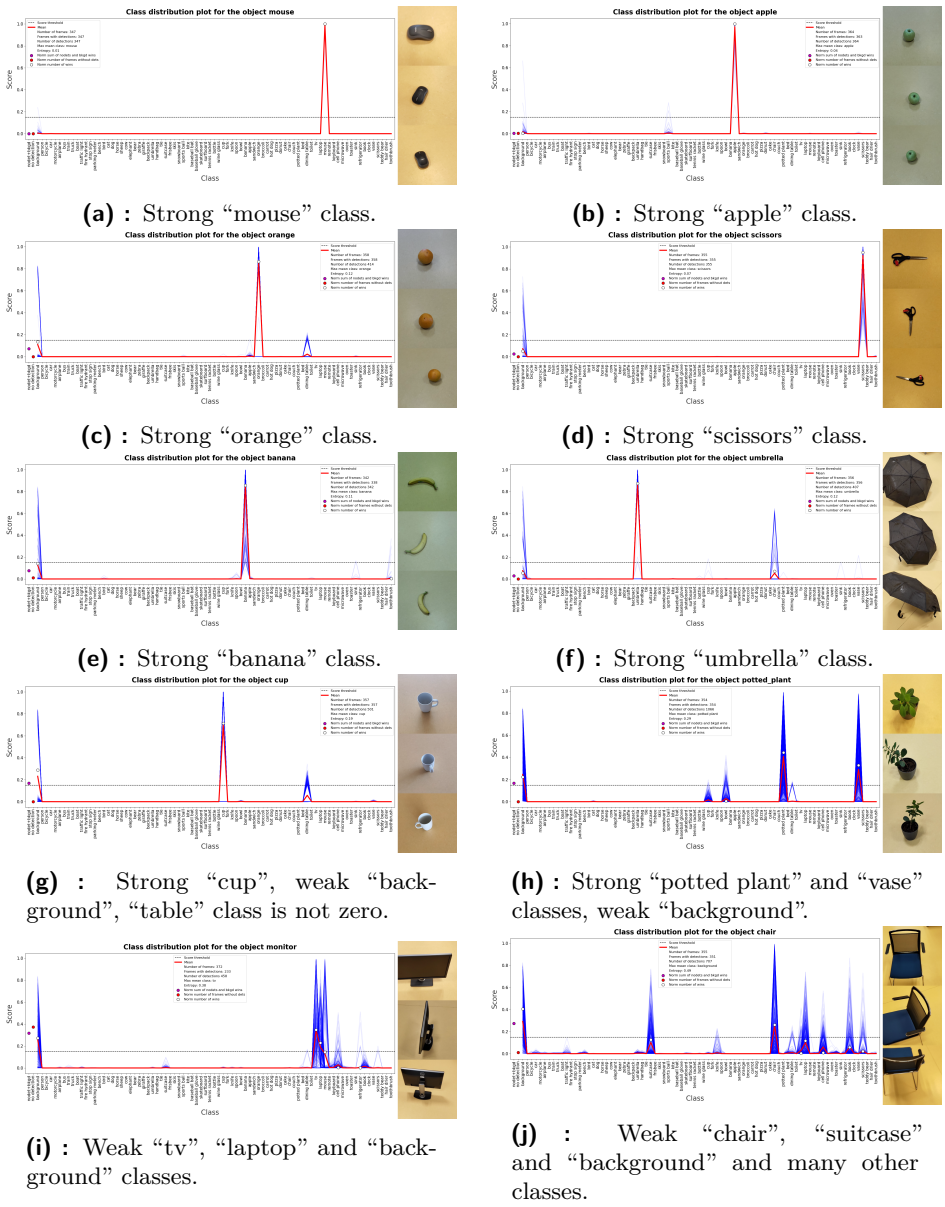
(i) : Strong “background”, weak “book” and “remote” classes.



(j) : Strong “background” but the number of detections is too small.

Figure 5.1: Test set. Known objects are on the left, unknown on the right. Blue lines correspond to the confidence scores of each detection. The red line corresponds to the mean confidence score through all detections. White points encode the normalised number of class wins.

Known objects discussion



(a) : Strong "mouse" class.

(b) : Strong "apple" class.

(c) : Strong "orange" class.

(d) : Strong "scissors" class.

(e) : Strong "banana" class.

(f) : Strong "umbrella" class.

(g) : Strong "cup", weak "background", "table" class is not zero.

(h) : Strong "potted plant" and "vase" classes, weak "background".

(i) : Weak "tv", "laptop" and "background" classes.

(j) : Weak "chair", "suitcase" and "background" and many other classes.

Figure 5.2: Training known set. Blue lines correspond to the confidence scores of each detection. The red line corresponds to the mean confidence score through all detections. White points encode the normalised number of class wins.

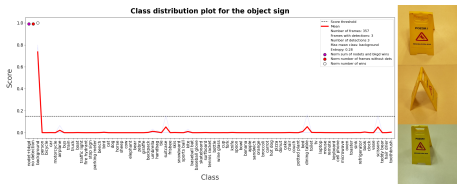
Figure 5.2 shows no doubt for some objects (mouse, orange, umbrella) that they are represented enough in the detector. Most of the distribution mass through all detections is concentrated in 1 correct class for the top known objects. The final decision can be made by simply taking a class from a single peak. An interesting case is represented by the "cup" object. Because in the

COCO dataset, all objects are placed in the natural context, in many images during training, the cups stand on a table. Probably, sometimes cups are annotated as part of the table. That is why "table" class confidence in many detections is not zero, which affects the mean score value.

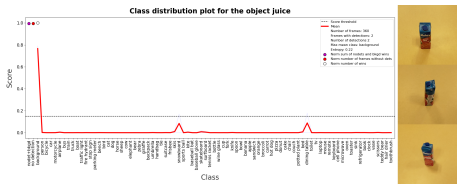
Some of the known objects are composite (laptop contains a keyboard, potted plant contains vase), leading to the presence of 2 competing peaks and the growth of the "background" class. The correct class is still winning, but consideration about more than 1 class is already needed. These correlations cannot be easily extracted for all possible object combinations manually. Instead, a solution that can learn it in an end-to-end manner should be introduced. Parts of the composite objects can be considered and studied separately, but in this case, more complex solutions that can accurately differentiate parts of the objects are needed.

Some objects like "skateboard" or "monitor" (which can be treated as tv) are detected as many different objects. It can be evidence that these objects are underrepresented in the training dataset. More labelled instances can be provided to the model to learn to detect confusing objects better. Alternatively, the same as for complex objects, we can extract the similarity correlations between detected classes and make the decision based on them. For example, a skateboard without wheels looks almost like a surfboard. That should be reflected in the final decision about the 3D object.

If our solution is based only on taking the class with maximum probability, the "book" and "chair" objects will be wrongly classified as "background". That happens because in many detections, confidence is low, so scores are distributed over the other classes and negative class. Although the winning "thing" classes are correct, the detector was not confident in its decisions.



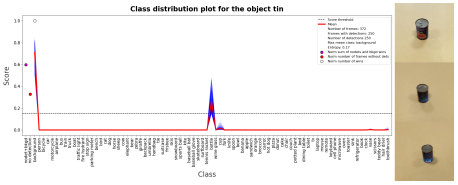
(a) : Strong "background" but the number of detections is too small.



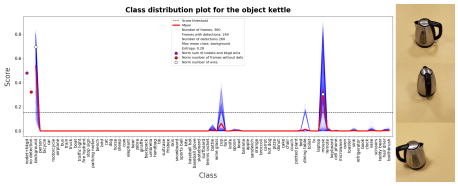
(b) : Strong "background" but the number of detections is too small.

Figure 5.3: Unknown objects with low number of detections. Blue lines correspond to the confidence scores of each detection. The red line corresponds to the mean confidence score through all detections. White points encode the normalised number of class wins.

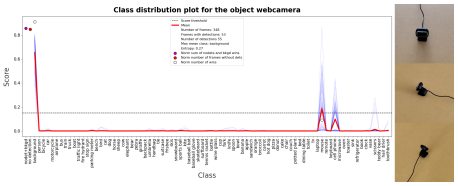
Unknown objects discussion



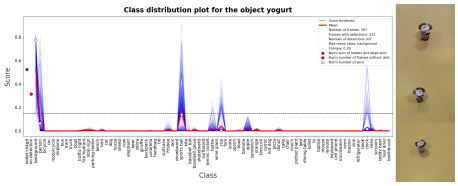
(a) : Strong “background” and weak “bottle”.



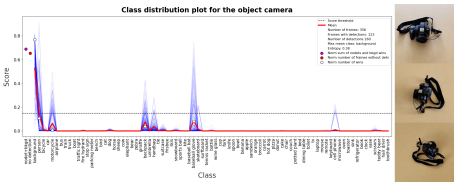
(b) : Strong “background” and weak “mouse”.



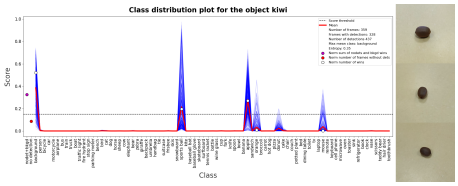
(c) : Strong “background”, weak “apple” and “sports ball”.



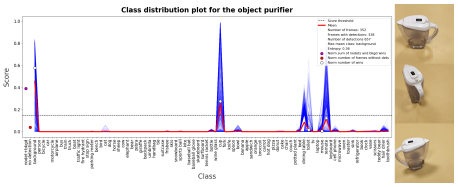
(d) : Strong “background” and weak “sports ball”.



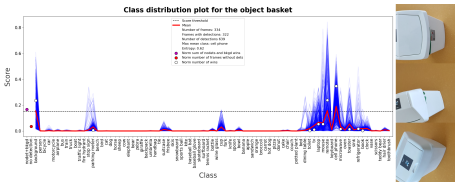
(e) : Strong “background” but the number of detections is small.



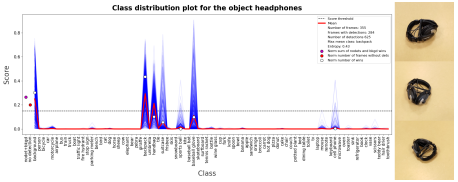
(f) : Strong “background”, weak “apple” and “sports ball”.



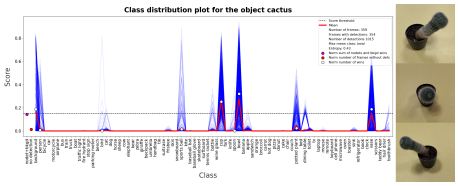
(g) : Strong “background”, weak “cup” and “remote”.



(h) : Weak “cell phone”, “remote” and “background” and many other classes.



(i) : Weak “backpack”, “handbag” and “background”.



(j) : Weak “bowl”, “cup”, “vase” and “background”.

Figure 5.4: Training unknown set. Blue lines correspond to the confidence scores of each detection. The red line corresponds to the mean confidence score through all detections. White points encode the normalised number of class wins.

The first difference between known and unknown is that the "background" class is very high in most cases for unknown objects (Figure 5.4). Objects like "tin", "yogurt" or "camera" have a very high score in "background". It seems that for many objects, simple thresholding on "background" will work well to classify unseen objects during training. The internal ability of the detector to reject objects which are not from trained classes might be enough.

Nevertheless, there are examples such as "basket", "headphones", "computer" with winning "thing" classes where there are much more peaks in comparison with knowns. Classes that correspond to these peaks do not correlate with each other well. Thus, the amount and the semantic difference between the most probable classes can serve as a good indicator of the unknown.

For some objects like "ukulele", "juice", and "sign" (Figure 5.3), the detector is insensitive. There are two reasons for this: 1) These objects were present in the training dataset, and the detector learned to reject them very well, so they are represented well as negative examples, 2) Detector did not learn to reject these objects, but their appearance is so different from any known class, that the detector can not even misclassify them as something else. The same happens to people. When someone does not even imagine the existence of some entity and has never seen anything similar to it, it is tough to notice this entity. In both cases, the insensitivity can be fixed by expanding the boundaries of knowledge. The more we know, the more we realise how much we do not know.

5.3 Computation of thresholds

Having the data described and visualised in the previous section, we can compute the thresholds for the methods described in 3.5.3. While we want to find a decision boundary between known and unknown, we can treat this problem as a binary classification task. We use SVM with a linear kernel, train it on the extracted 1D features and take only the final hyperplane as threshold. For testing, we use 10 objects transferred to the simulation and train on the remaining 20 objects (except "sign" and "juice" due to the small number of detections). The number of knowns and unknowns is equal. Although the dataset size is insufficient to train a decent model, we find that the obtained results show that our approach may serve as a good baseline. We note that much more data is needed to get better estimates of the learned parameters. Decision boundary and train values for both approaches are in Figure 5.5. Although we separately consider the "background" class in the

final decision using entropy values, in this section, for training and testing, we rely only on entropy values. The reason is the small size of the dataset.

Figure 5.5 shows that there is a correlation between the extracted features and the fact that the object is known or unknown. However, the entropy values are not much higher for unknown objects. Both our methods do not consider the similarity of the classes. That is why the entropy value will be the same for 2 different cases: 2 strong peaks for classes from the same subcategory ("tv" and "laptop" for "monitor" object) and 2 strong peaks from different subcategories ("sports ball" and "apple" for "kiwi" object). The entropy of the mean score distribution vector oversimplifies the problem.

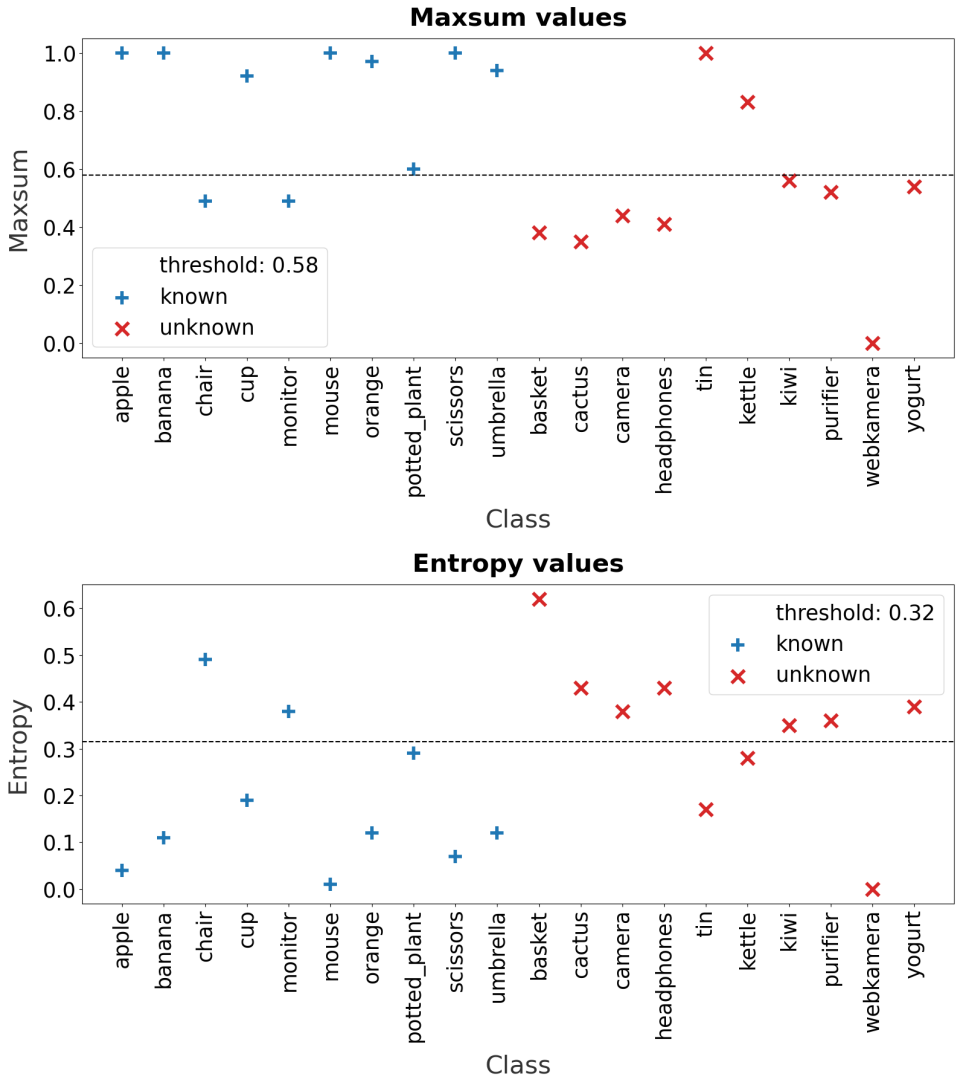


Figure 5.5: Maxsum and entropy decision boundaries on the train set. For maxsum, points above the thresholds are classified as known. For entropy, points under the threshold are classified as known.

To display the performance of our classifiers on the dataset, we plot ROC curves. On the training data, both classifiers perform relatively well according to the AUC scores (Figure 5.6). Maxsum method performs slightly better, possibly because now the "background" class is not considered for the entropy method. The entropy of unknown objects which have a strong "background" class in the distribution is low (e.g. "tin" and "webcamera"). However, performance on the test set is poor, as we observe from the low AUC scores (Figure 5.7), which is caused by the small size of the dataset and the simplicity of the proposed classifiers.

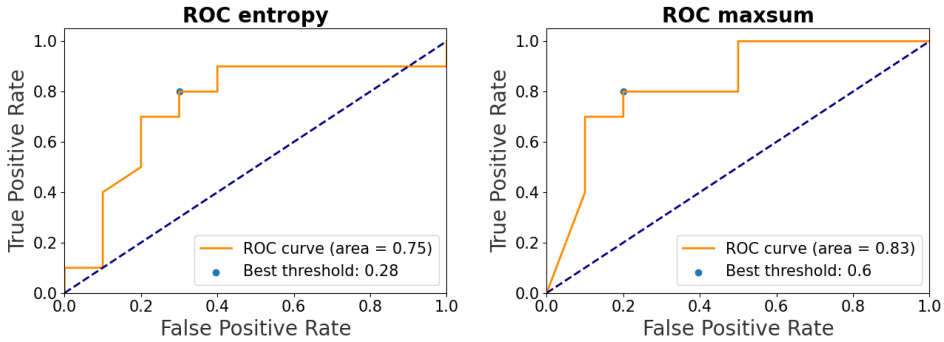
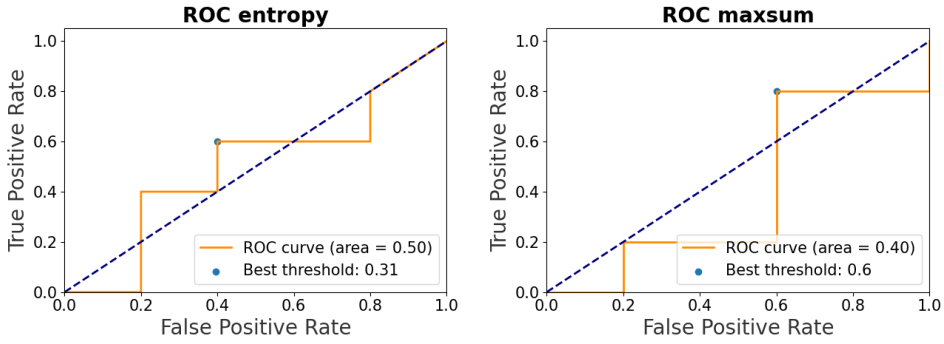
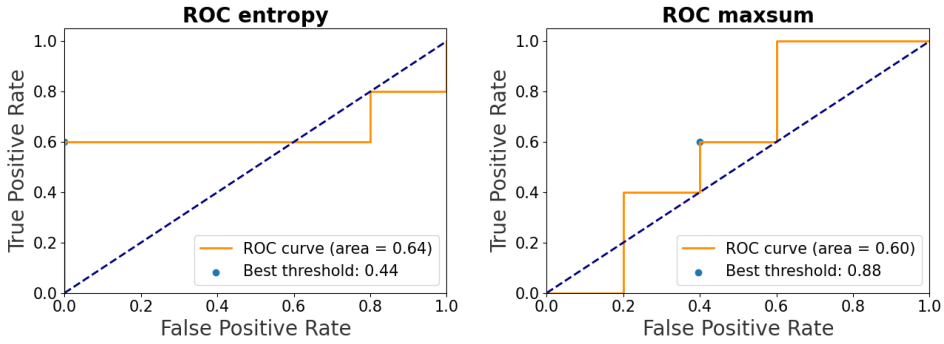


Figure 5.6: ROC for the train set.



(a) : Real objects.



(b) : Simulated objects.

Figure 5.7: ROC curves for the test data.

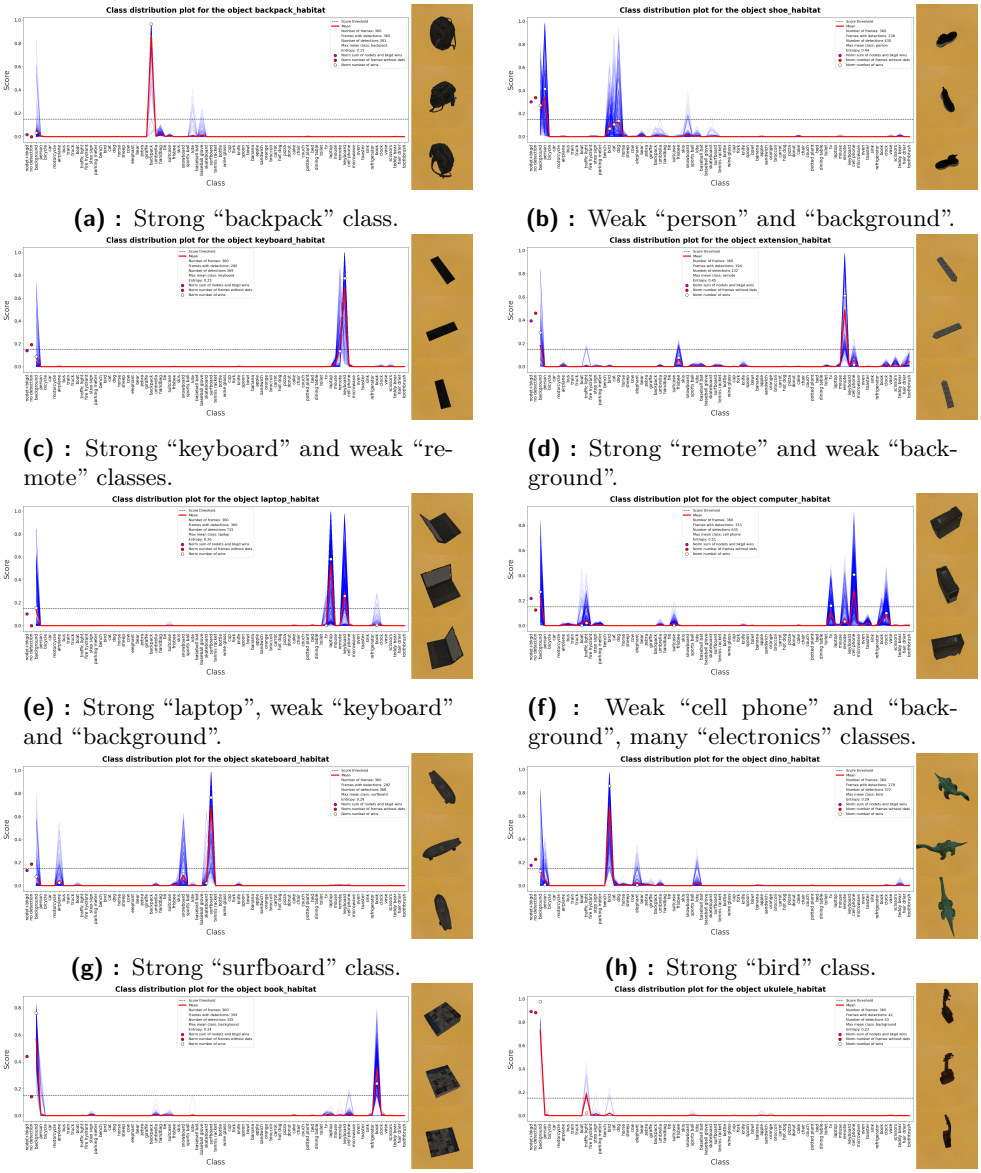
Table 5.1 shows no considerable difference between the results on the real test objects. Interestingly, there is a difference between the results on real and simulated objects for the entropy method. The difference in object structure or light conditions caused the increase of entropy in the score distributions for unknown objects. According to the results, entropy measuring for simulated objects is a better choice.

Test set and Method	Precision	Recall	F1 score	Accuracy
Real maxsum	0.62	0.60	0.58	0.60
Real entropy	0.60	0.60	0.60	0.60
Habitat maxsum	0.78	0.60	0.52	0.60
Habitat entropy	0.86	0.80	0.79	0.80

Table 5.1: Test results

5.4 Real and simulated object comparison

In the same way as for real objects, we produce videos for the objects transferred to the Habitat simulation. In the same manner, we can visualise the class distribution score through all frames (Figure 5.8).



(a) : Strong "backpack" class.

(b) : Weak "person" and "background".

(c) : Strong "keyboard" and weak "remote" classes.

(d) : Strong "remote" and weak "background".

(e) : Strong "laptop", weak "keyboard" and "background".

(f) : Weak "cell phone" and "background", many "electronics" classes.

(g) : Strong "surfboard" class.

(h) : Strong "bird" class.

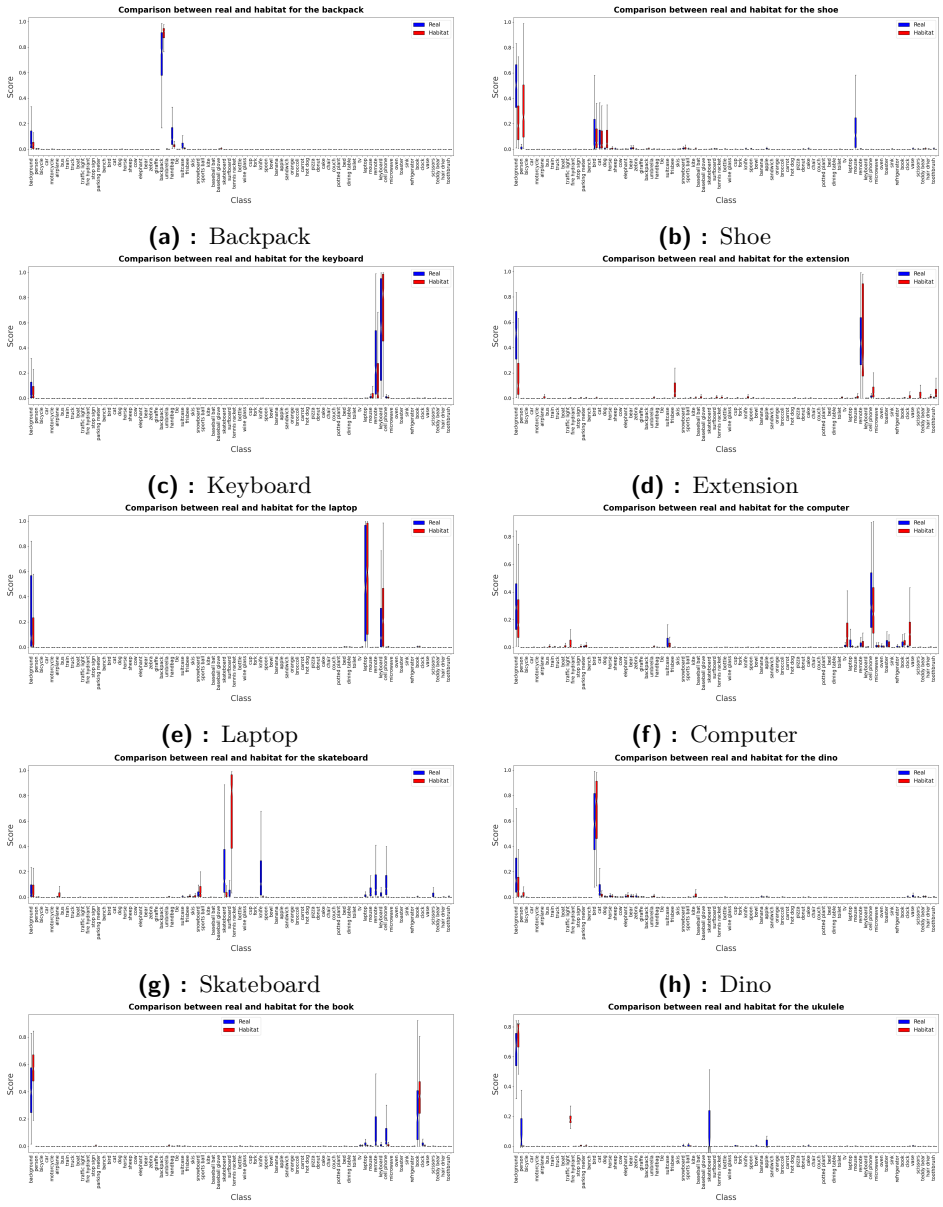
(i) : Strong "background" and weak "book" classes.

(j) : Strong "background" but the number of detections is too small.

Figure 5.8: Simulated test set. Known objects on the right, unknown on the left. Blue lines correspond to the confidence scores of each detection. The red line corresponds to the mean confidence score through all detections. White points encode the normalised number of class wins.

Moreover, we can compare the class probability distributions for both real and simulated objects. We use boxplots to visualize the main properties of detections received from YOLACT (Figure 5.9). According to the data on boxplots, the score distributions are very similar in both worlds for many objects. Therefore, sim2real transfer of the obtained model seems possible.

However, some real/simulation object pairs differ a lot due to minor differences in object appearance. For us, this similarity level is enough to show how our 3D open set object detection approach works in simulated but close-to-real scenarios.



(a) : Backpack **(b) :** Shoe

(c) : Keyboard **(d) :** Extension

(e) : Laptop **(f) :** Computer

(g) : Skateboard **(h) :** Dino

(i) : Book **(j) :** Ukulele

Figure 5.9: Real and simulated objects boxplot comparison. Known objects are on the left, unknown on the right. Blue is for real objects, red for simulated.

5.5 Open-Set 3D Object Detection

Now that we have insights about known and unknown differences, thresholds for the proposed methods and the confidence that the obtained results will not differ a lot in the real world, we can evaluate our approach in the simulation. Objects are placed at the RGB sensor level on stands (Figure 5.11a). The trajectory of camera movement inside the simulated environment is made manually so that every object is observed from many different viewpoints (Figure 5.10).

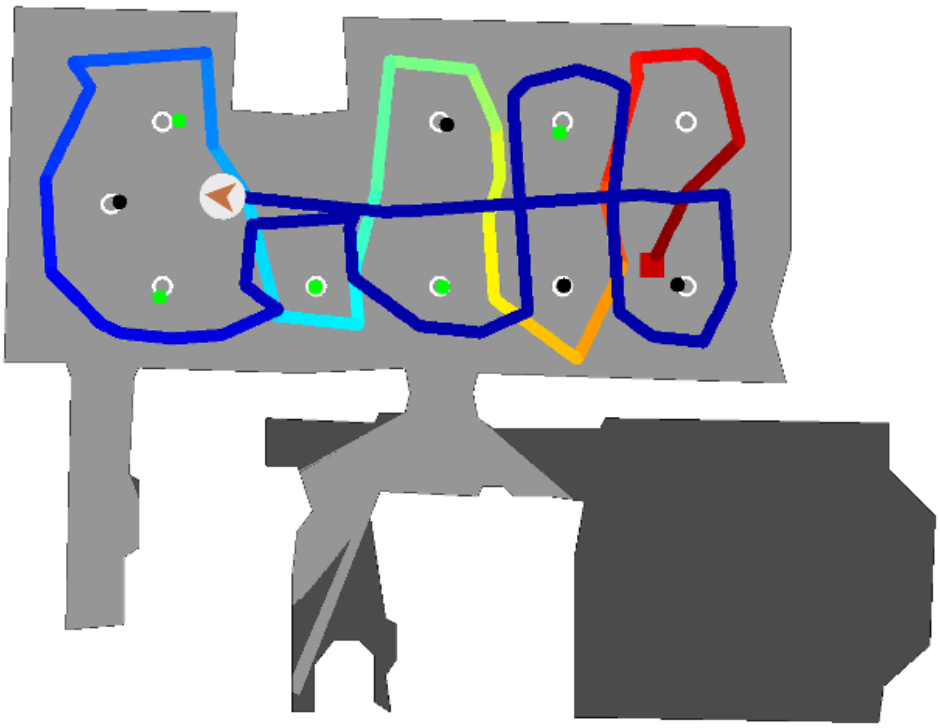


Figure 5.10: A trajectory of observer movement used for evaluation. The trajectory is projected onto a top-down map of the indoor space. The red square is a start position, arrow in a circle is the current agent position. White circles show the ground truth position of all added objects. Black circles stand for detected 3D objects centres and classified as "unknown", green ones decode objects classified with known labels.

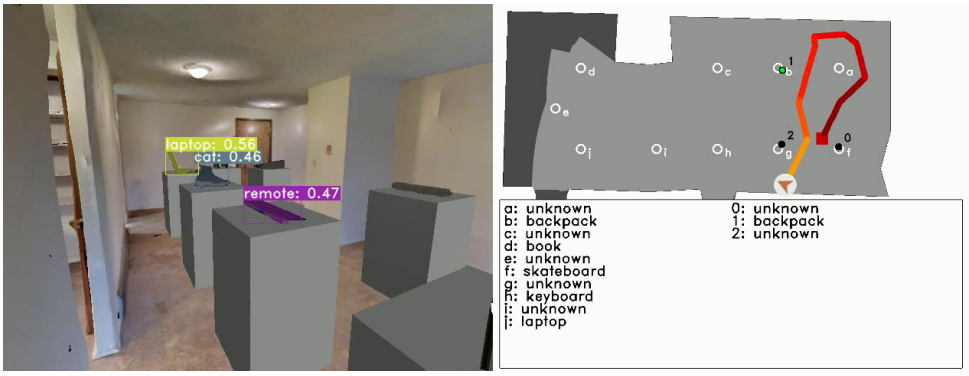
There are hyperparameters that we use to configure our method:

- **unknown threshold** - decision threshold for maxsum (0.58) or entropy (0.32)

- **yolact threshold** - YOLACT confidence threshold (0.3)
- **cluster radius** - the radius of the sphere that represents an object (0.5m)
- **seen threshold** - number of times an object is seen to make the first classification (50)
- **seen ratio** - ratio of the number of detections to the number of times object was seen (0.15)

We conduct experiments for both decision making approaches described in 3.5.3. Observer moves around the room with a predefined trajectory (Figure 5.10). YOLACT processes images in each timestep and outputs detections if any object detection has a confidence higher than **yolact threshold**. These detections are then assigned to clusters with radius **cluster radius** which represent 3D objects. When object is captured on RGB sensor more than **seen threshold** times and **seen ratio** is high enough, we classify the object using one of the approaches. Low **seen ratio** means that we receive detections in the some place rarely and, most probably, that are false positives. The main parts of this process are illustrated in Figure 5.11. At the end of the episode, we have all detected objects with assigned labels from the set $S = \{0\} \cup K$. Detected objects are compared with ground truth object information, i.e. object 3D position and label.

(a) : YOLACT detects 1 object, no 3D objects are detected yet.



(b) : YOLACT detects 3 objects and few 3D object are already detected.

(c) : Almost all 3D objects are detected, white label is the class associated with 3D position.

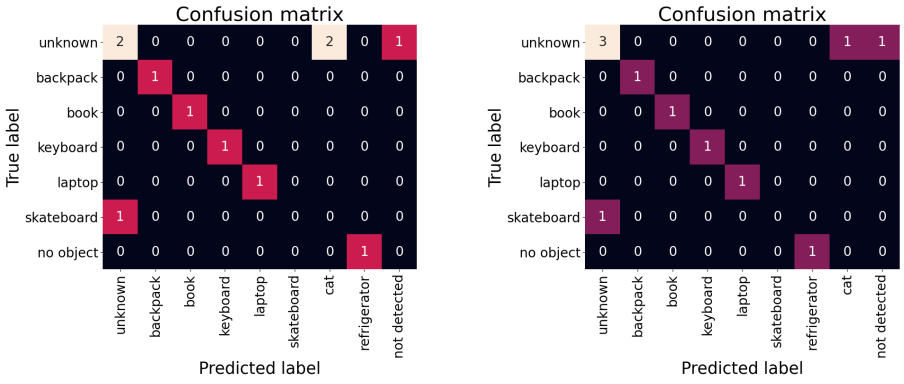
Figure 5.11: Example images from RGB sensor with corresponding top-down maps. Images captures 3 different phases: a)initial phase, b)middle phase and c)final phase.

From Table 5.2 it can be noted that entropy performs better than maxsum. All decisions except object "dino", which was classified as "unknown" by entropy and "cat" by maxsum, are the same. The top-down view of the detected objects and their labels for the entropy method is in Figure 5.11c. Figure 5.12 shows that both approaches can classify unknown objects, pre-

serving the ability to classify known ones. Both approaches have problems with the object "shoe", which YOLACT almost all the time detects as "cat" with high probability. Object "skateboard" has an unusual appearance and is not represented in the COCO dataset enough, so YOLACT classifies it with many different labels from different views. The unknown object with the label "ukulele" was not detected at all. That happened because this object has such an appearance that it is very different from any known object. This type of unknown cannot be detected with our approach. However, we expect that with an increasing number of known classes, the sensitivity to unknown objects will also increase.

Moreover, our method detected the wall and classified it as a "refrigerator". There were more "refrigerators" detected during the whole trial, but they disappeared with time because **seen ratio** was too small.

Object "dino" has very confusing labels of different animals ("cat", "bird", "elephant"), which is interesting because, even though this class was not present in the dataset on which YOLACT was trained, we can treat this object as an "unknown" animal. We chose "dino" to retrain YOLACT.



(a) : Maxsum **(b) :** Entropy

Figure 5.12: Detection results. Unknown objects are merged into 1 unknown class.

Method	Precision	Recall	F1 score
maxsum	0.5	0.5	0.5
entropy	0.57	0.57	0.57

Table 5.2: Evaluation results

5.6 YOLACT retraining

After identifying an unknown object and getting response from the oracle with a true label, we can extend our set of known classes. Our approach is straightforward:

1. Generate new dataset as described in Section 4.2.3
2. Extend COCO dataset with newly generated dataset
3. Train YOLACT on the extended dataset from scratch

We use the same hyperparameters as described in [3]. As a backbone CNN for extracting visual features, we use ResNet-101 [16]. Table 5.6 shows no considerable difference between the mAP scores of the original model and the retrained one. The old classes were not forgotten while the new class was learned. Interestingly, the mAP score on validation of the new class is 97.29, while the highest score between COCO classes is 61.31 for the class "bear".

Although such a training approach is scalable in terms of new classes, it is computationally heavy - the learning process takes a few days. More intelligent and more efficient solutions should be used in the future work. Our approach can serve as a proof of concept.

Training set	val2017	val2017 + "dino"	test-dev
train2017	28.47	27.85	28.40
train2017+"dino"	28.68	29.53	28.90

5.7 Performance with retrained YOLACT

To make sure that adding a new class works as it should, we conduct the same experiments as described above using a retrained detector. Results of the score distributions on different views of the objects extracted from the video are in Figure 5.13. Both in real and simulated conditions, the detector successfully identifies a new object as "dino". However, the "background"

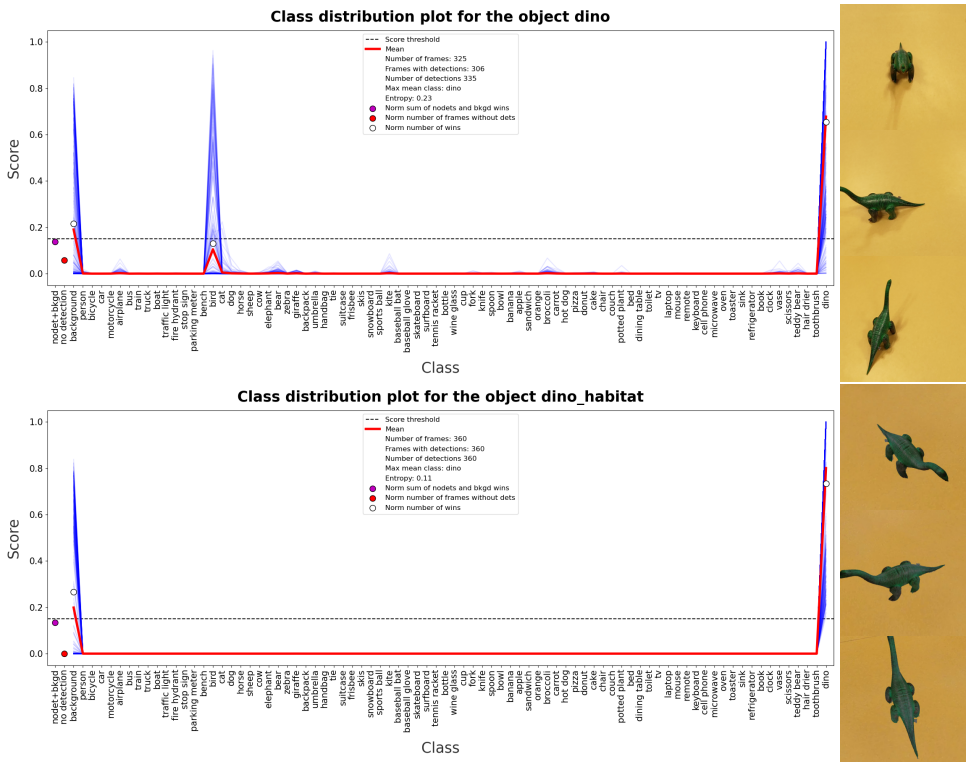


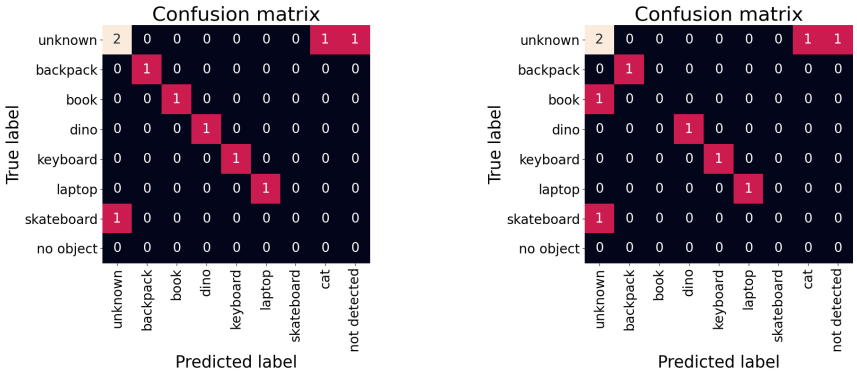
Figure 5.13: Class probability distributions for real and simulated object that was used for retraining. Strong new "dino" class and "weak" background in both cases.

component is not zero, so there is still room for improvement in the dataset generation process.

Additionally, we study how the retrained detector works as a part of the whole approach pipeline. We evaluate retrained YOLACT on the walk around the indoor space. Figure 5.14 shows that a new object is detected and classified correctly in both approaches and the other detections are the same as for the basic model. However, the entropy approach failed to classify the object "book" correctly. Quantitative results are in the Table 5.3. For the maxsum method, all metrics are higher, because as we mentioned above, we do not count the correct detections of the unknown when computing precision and recall. Thus, the open set F1 score is higher when 1 of the unknown objects become known, and we detect it correctly, as it should be.

Method	Precision	Recall	F1 score
maxsum	0.71	0.63	0.67
entropy	0.57	0.50	0.53

Table 5.3: Evaluation results with retrained YOLACT



(a) : Maxsum **(b) :** Entropy

Figure 5.14: Detection results with retrained YOLACT.

Figure 5.15 shows detections made in the same place and time during the evaluation trial. Retrained YOLACT outputs correct detection with high confidence on the object "dino".

Figure 5.15: YOLACT outputs on the same frame in basic version on the left and retrained version on the right.

5.8 Results discussion

In our experiments, we show that our approach is reasonable but still needs improvements in each component. From the distributions we obtained for known and unknown objects, we see that unknown objects have more confusing labels and the background component is relatively high. Maxsum and entropy of the mean score distribution vector perform well but do not fully correspond to the distributions we receive. We do not extract semantic similarity to use it for known/unknown classification, which is crucial in our problem setting. Evaluation in the simulation seems to be a good approximation of the performance in the real world. The process of learning a new class succeeded

on both real and simulated data, even though the dataset was generated in the simulation.

The test object set, which we transfer to the simulation, does not contain the best representatives of known and unknown objects in terms of stability of the produced detections. Nevertheless, this set better corresponds to the real world, where there are unknown objects which are undetectable ("ukulele") or too similar to another object ("shoe" is similar to "cat" for the detector) and known objects that are not represented enough in the detector ("skateboard" or "book").

Overall, our method can serve as a baseline, and our evaluation protocol can be used for testing more complex and accurate systems. We have found out that such systems should take the correlations between outputs into account.

Chapter 6

Conclusion and perspectives

6.1 Summary

In this work, we formalise open set 3D object detection and provide a solution that combines existing approaches from different research fields. We consider the open set setting from a new perspective, trying to exploit the understanding of 3D objects. We show that detections made from different viewpoints may serve as a good indicator of newness. To evaluate such a system, we create a simulation with close-to-real conditions and conduct controlled experiments. We demonstrate that it is possible to develop a model that can identify unknown objects while preserving the ability to correctly assign labels to classes present during training. Certainly, more complex models can be developed to solve this problem, and our pipeline can speed up this process. Finally, we demonstrate how the process of learning new classes can be carried out. Such a learning process takes a long time, although it is effective, as we demonstrated. We make a step to develop the ability to recognise unknown objects, which will improve robots' understanding of the environment and make their decisions more accurate and safe.

6.2 Future work

Better decision making. The similarity of predicted classes for one object captured in different conditions is a good indicator of whether an object is unknown. We plan to develop a more complex model capable of understanding deeper relationships between objects, classes and sequences. The possible solution may be based on transformers [45], a popular architecture for sequence prediction. Transformers are usually used in Natural Language Processing tasks. However, some works adapt transformers for the Computer Vision field. For example, DETR [6] is an object detection framework based on transformers. DETR is competitive and conceptually simple at the same time.

Dynamic environment. In our work, we assumed that all objects in the environment are static so that they can be identified by their absolute 3D coordinates in some coordinate system. However, what if the object can move? How to represent it then? Furthermore, how to be sure that the object the robot saw last time 5 minutes ago in the same place is the same object? Many of these problems can be solved using 3D Dynamic Scene Graphs [40]. Nevertheless, further research of object representation in dynamic environments is still needed.

Better object representation. Our representation of objects as spheres with fixed radius is unrealistic. We need a clustering method capable of dealing with objects of all possible shapes and sizes. For example, it can be bounding boxes or point clouds extracted from semantic masks.

Speeding up the detector. Although the detector used in our work is fast, the total speed is too low for the real applications in combination with other parts of our approach. Therefore, we need to use some of the modifications of YOLACT or find an alternative. When our system is fast enough, the next logical step will become possible - transferring the model to the real world.

Appendix A

Bibliography

[1] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[2] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019.

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact++: Better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[5] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[7] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[8] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[9] Terrance DeVries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks, 2018.

[10] Akshay Dhamija, Manuel Gunther, Jonathan Ventura, and Terrance Boult. The overlooked elephant of object detection: Open set. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[12] ZongYuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. Generative openmax for multi-class open set classification, 2017.

[13] Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2020.

[14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[17] K J Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection, 2021.

[18] Pedro R Mendes Júnior, Roberto M De Souza, Rafael de O Werneck, Bernardo V Stein, Daniel V Pazinato, Waldir R de Almeida, Otávio AB Penatti, Ricardo da S Torres, and Anderson Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[21] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks, 2020.

[22] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[25] Haotian Liu, Rafael A. Rivera Soto, Fanyi Xiao, and Yong Jae Lee. Yolactedge: Real-time instance segmentation on the edge, 2021.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[27] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[28] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[29] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[30] Dimity Miller, Feras Dayoub, Michael Milford, and Niko Sünderhauf. Evaluating merging strategies for sampling-based uncertainty techniques in object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2348–2354, 2019.

[31] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sünderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3243–3249, 2018.

[32] Dimity Miller, Niko Sunderhauf, Michael Milford, and Feras Dayoub. Class anchor clustering: A loss for distance-based open set recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3570–3578, January 2021.

[33] Dimity Miller, Niko Sünderhauf, Michael Milford, and Feras Dayoub. Uncertainty for identifying open-set errors in visual object detection, 2021.

[34] Pramuditha Perera, Vlad I. Morariu, Rajiv Jain, Varun Manjunatha, Curtis Wigington, Vicente Ordonez, and Vishal M. Patel. Generative-discriminative feature representations for open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[35] Federico Pernici, Federico Bartoli, Matteo Bruni, and Alberto Del Bimbo. Memory based online learning of deep representations from video streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] Stanislav Pidhorskyi, Ranya Almohsen, Donald A Adjeroh, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders, 2018.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[39] Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of ncm forests for large-scale image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[40] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans, 2020.

[41] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[42] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[43] Walter J. Scheirer, Lalit P. Jain, and Terrance E. Boult. Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, 2014.

[44] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[46] Michal Vavrecka, Nikita Sokovnin, Megi Mejdrechova, Gabriela Sejnova, and Marek Otahal. mygym: Modular toolkit for visuomotor robotic tasks, 2020.

[47] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[48] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.