POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Towards Temporal Consistency in Egocentric Object Detection for Open-Vocabulary Navigation

Supervisors

Prof. Giuseppe Bruno AVERTA

Ing. Marco CICCONE

Ing. Claudia CUTTANO

Ing. Gabriele TIBONI

Candidate

Antonio DE CINQUE

April 2024

Abstract

Egocentric object detection is a critical aspect of robotic navigation and interaction within dynamic and complex home environments. The primary objective of this research is to explore the challenges and solutions associated with achieving temporal consistency in egocentric object detection, particularly in the scope of the Open-Vocabulary Mobile Manipulation (OVMM) challenge. This is contextualized within the HomeRobot 3D simulation environment, where a robot (Hello Robot Stretch) is tasked with navigating a household and bring an object from one place to another. The perception module of the robot is enabled by open-vocabulary object detection models, such as DETIC (Detecting Twenty-thousand Classes using Image-level Supervision). These models have shown to be promising in recognizing a wide range of objects, given any prompt. However, the performance is often hindered by the egocentric view and the lack of a temporal coherence, leading to "noisy" predictions and inconsistencies across consecutive frames. This problem arises as the model is processing each frame in isolation, leading to predictions which lack continuity and coherence across time. To address this, we investigate the integration of Spatio-Temporal Adapters (ST-Adapters) within the DETIC model, aiming to enhance the model's ability to maintain temporal consistency, without compromising its open-vocabulary capabilities. We highlight the limitations of DETIC in maintaining temporal consistency, particularly in the detection of small or partially obscured objects. By incorporating ST-Adapters, we investigate an approach to instill the model with a spatio-temporal dimension, allowing for more coherent and reliable object detections over time. The HomeRobot simulation environment leverages a subset of the Habitat Synthetic Scenes Dataset (HSSD), featuring high-quality 3D scenes. For our analyses, we extracted frames from video explorations, complete with objects' ground truths. To assess the performance of the models, we conduct evaluations using two distinct test sets: one in-domain, consisting of scenes similar to those encountered during the training phase, and another out-of-domain, comprising household scenes not seen in training. Therefore, the hypothesis to verify is that an improvement in the frame detections would translate into enhanced temporal consistency in the 3D simulation environment.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The advent of household robotic assistants represents a pinnacle of achievement and a continuous source of inspiration in the field of robotics. This enduring ambition has catalyzed significant strides across multiple research domains within robotics, encompassing **vision**, **manipulation**, and the integration of complex tasks and benchmarks. The ultimate goal is to develop a robot that is not only proficient in recognizing a vast array of objects but also capable of interacting with the environment and navigating intelligently in a world where sensory information is inherently limited.

This broad spectrum of research has driven advancements in various related fields, including **navigation** [1, 2], **service robotics** [3, 4], **language understanding** [5, 6] and the integration of **task and motion planning** [6].

These endeavors are unified under the challenge of **Open-Vocabulary Mobile Manipulation (OVMM)**. The essence of OVMM lies in the development of a robot capable of identifying and relocating arbitrary objects within the dynamic and unpredictable confines of a home environment.

1.2 Open-Vocabulary Mobile Manipulation

The concept of **Open-Vocabulary Mobile Manipulation (OVMM)** is an advanced directive in robotics that combines three significant aspects to address the challenges of operating within home environments:

- **Open-Vocabulary**: the robot's ability to recognize and understand a wide array of objects without being limited to a pre-defined set of items. In traditional settings, robots are programmed to identify and interact with specific objects they have been trained on. However, an **open-vocabulary** approach enables the robot to adapt to new and unforeseen objects, enhancing its functionality in real-world scenarios. This capability is crucial in a home environment, where the variety of objects can be vast. The "open-vocabulary" aspect allows the robot to process and act upon commands involving any object it encounters, leveraging advanced machine learning and natural language processing techniques.

- **Mobile**: the robot's capability to autonomously navigate and move across different areas within the home. Unlike stationary robots, a **mobile** robot can traverse various terrains and obstacles, such as stairs or doorways. This mobility is fundamental for performing tasks across multiple rooms or areas, enabling the robot to efficiently go where its assistance is required. Technologies like SLAM (Simultaneous Localization and Mapping) are integral, allowing the robot to understand, map its environment in real-time, and navigate effectively.

- **Manipulation**: the robot's ability to physically interact with objects by moving them from their initial location (*start_receptacle*) to a designated target location (*goal_receptacle*). This requires not only the robot's capability to identify and grasp various objects but also to understand the most appropriate methods for transporting these items safely and efficiently to their intended destinations. Manipulation, therefore, involves precise control and coordination, underlining the robot's ability to assist with a broad range of household tasks through direct physical interaction with the environment.

Combining these elements, **Open-Vocabulary Mobile Manipulation** represents a comprehensive approach to developing robotic systems capable of autonomously performing a wide range of tasks within the unpredictable and dynamic environments of human homes. OVMM robots are designed not only to understand and interact with any object and move freely throughout the space but also to manipulate items effectively, making them invaluable for advancing home automation and assistance.

1.2.1 The Modules of OVVM

Historically, research in this area has often simplified the context significantly, employing discrete action spaces, limiting the diversity of objects, or confining the scope to small, single-room environments that can be easily navigated. These constraints have hindered the application of these advancements in large, continuous, real-world settings. Nevertheless, recent breakthroughs in bridging **language and vision**, primarily through multi-modal models like **CLIP** [7], have broadened the horizon for robots to generalize across diverse object categories, surpassing traditional limitations.

A pivotal aspect of this research is the challenge posed by the integration of three main modules: **perception, planning, and action**. The perception module is responsible for recognizing and segmenting objects, the planning module for generating a sequence of actions to reach the target object, and the action module for executing these actions.

1.2.2 The Role of Perception in OVMM

In the perception module, DETIC plays an essential role: providing object segmentation inputs for both heuristic and Reinforcement Learning (RL) navigation strategies. This means that the goal of creating versatile and effective general-purpose home assistants necessitates a comprehensive approach that deeply connects the elements of perception, planning, and action. In this work, however, our focus narrows to the perception aspect, recognizing it as the foundational step in this intricate pipeline. By concentrating on enhancing perception, we lay the groundwork for subsequent improvements in planning and action.

1.3 Problem Statement

The development of robotic assistants for household environments requires the integration of advanced capabilities in object detection, navigation, and interaction within dynamic and unpredictable settings. A critical challenge in this domain is achieving reliable and consistent object detection in an egocentric perspective, which is essential for the effective functioning of such robots. This challenge is amplified in the context of Open-Vocabulary Mobile Manipulation (OVMM), where the robot must identify and interact with a wide variety of objects within complex home environments.

Despite significant advancements in robotics, particularly in the areas of vision, manipulation, and task integration, the specific issue of **temporal consistency** in egocentric object detection remains a notable gap. This inconsistency arises due to the limitations of current perception models, such as DETIC, which, despite their ability to recognize a broad range of objects, struggle with maintaining continuity across consecutive frames. This results in "noisy" predictions and hampers the robot's ability to interact effectively with its environment, especially when dealing with small or partially obscured objects.

1.4 Objectives

The primary objectives of this research are outlined as follows:

1. **To Understand the Current Limitations of Egocentric Object Detection:** Examine the existing challenges in achieving temporal consistency in object detection, particularly in the context of Open-Vocabulary Mobile Manipulation (OVMM). This involves a comprehensive review of the DETIC model and its performance in dynamic and complex home environments.

2. **To Investigate the Integration of Spatio-Temporal Adapters (ST-Adapters):** Explore the potential of incorporating ST-Adapters into the DETIC model to enhance its ability to maintain temporal consistency. This includes understanding how spatio-temporal dimensions can be leveraged to achieve more coherent and reliable object detections across consecutive frames.

3. **To Evaluate the Performance of Enhanced Object Detection Models:** Conduct testing and evaluation of the modified DETIC model with ST-Adapters in the HomeRobot 3D simulation environment. This involves comparing the model's performance on two distinct test sets: one in-domain (scenes similar to the training phase) and another out-of-domain (unseen household scenes).

4. **To Assess the Impact of Improved Temporal Consistency on Robotic Navigation and Interaction:** Analyze how enhancements in object detection temporal consistency affect the overall capabilities of robots in navigating and interacting within home environments. This includes exploring the implications for subsequent modules of planning and action in the robotic operation.

Chapter 2

Literature Review

2.1 Egocentric Object Detection

Egocentric object detection (**EOD**) is a challenging task that requires the detection and localization of objects from a first-person perspective. This task is particularly relevant in the context of home robotics, where robots must understand and interact with objects in their environment. However, the unique challenges posed by EOD have made it difficult to develop robust and accurate detection systems.

While progress has been made in computer vision, particularly in naming objects and activities in Internet photos or video clips, current systems and datasets represent a limited definition of visual perception. Today's models excel in interpreting isolated moments from a third-person "spectator" view, fueled by significant dataset and benchmark efforts. However, the realm of robotics demands an understanding from a first-person or "egocentric" perspective, where the input is a long, fluid video stream that captures the world through the eyes of an actively engaged agent. Unlike Internet photos, captured intentionally by a human photographer, egocentric footage from an always-on wearable camera lacks active curation. Moreover, first-person perception requires a persistent 3D understanding of the wearer's physical surroundings and an interpretation of objects and actions within a human context. This necessitates attention to human-object interactions and high-level social behaviors, pushing the boundaries of how we define and tackle visual perception in computer vision.

2.1.1 Challenges in EOD

Egocentric object detection (**EOD**) presents unique vision challenges that stem from the nature of first-person perspective imaging. These challenges significantly impact the development of robust and accurate detection systems. Key vision-related challenges include:

- Dynamic environment changes, such as **varying lighting conditions, clutter, and occlusion**, significantly affect the detection quality.
- The **egocentric perspective** introduces challenges such as motion blur and scale variation due to the camera's movement, and a limited field of view that captures only a portion of the scene from the robot's or user's perspective.
- The reliance on third-person data for pre-training egocentric video models introduces challenges due to a **sizeable domain mismatch** between the training datasets and real-world egocentric applications. Large-scale annotated datasets such as Kinetics [8], AVA [9], UCF [10], ActivityNet [11], HowTo100M [12], ImageNet [13], and COCO [14] focus primarily on third-person Web data, exhibiting a significant domain mismatch when applied to egocentric video understanding.

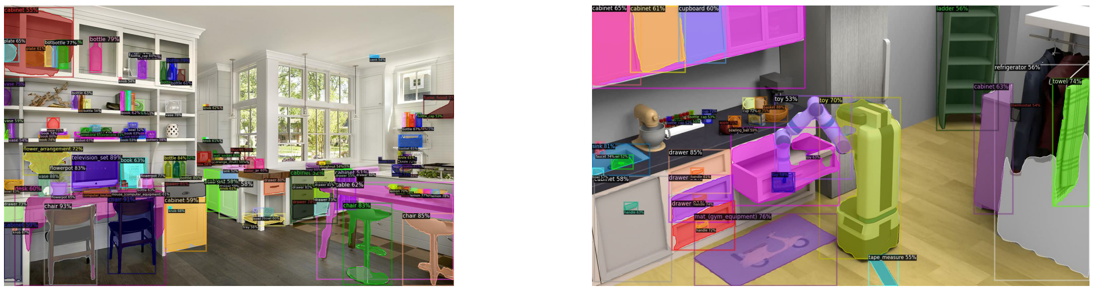


Figure 2.1: Left: Detecting objects in a real-life environment. Right: Detecting objects in a 3D simulated environment (AI Habitat).

Beyond the immediate vision challenges, egocentric object detection (EOD) encompasses a broad spectrum of challenges that extend into robotics and human-computer interaction. These include:

- **Human-object interactions**: understanding how humans interact with objects in their environment, which is critical for robots to assist or cooperate with humans in daily tasks. It involves recognizing not just the objects but also the purpose and nature of the interaction.

- **Activity recognition**: identifying the activities being performed by humans from the egocentric perspective. Recognizing activities can help in understanding the context of the scene and predicting future actions, which is crucial for robots to interact appropriately in human environments.
- **Anticipation**: predicting future actions or events before they occur. This is especially important in dynamic environments where robots need to plan their actions in advance to avoid obstacles or assist humans proactively.
- **Inferring the camera wearer's body pose**: estimating the posture and orientation of the person wearing the camera, which is crucial for understanding the person's current activities and intentions. Accurate pose estimation can enhance the robot's understanding of the user's actions and improve interaction quality.

The expansion of egocentric datasets and the evolution of machine learning techniques, including the integration of temporal information and spatial context, signify the ongoing efforts to overcome the inherent challenges of EOD. These advancements are critical for the development of robust and accurate egocentric object detection systems that can be effectively deployed in home robotics and other real-world applications.

2.1.2 Open-Vocabulary Object Detection

Open-vocabulary object detection, also known as **zero-shot object detection**, represents a significant leap forward in the field of computer vision, aiming to identify and localize objects that fall outside the predefined categories encountered during model training. Traditional object detection models are limited by their training datasets; they can only recognize objects they have been explicitly trained to identify. In contrast, open-vocabulary models break this constraint by leveraging language embeddings to bridge the gap between seen and unseen objects.

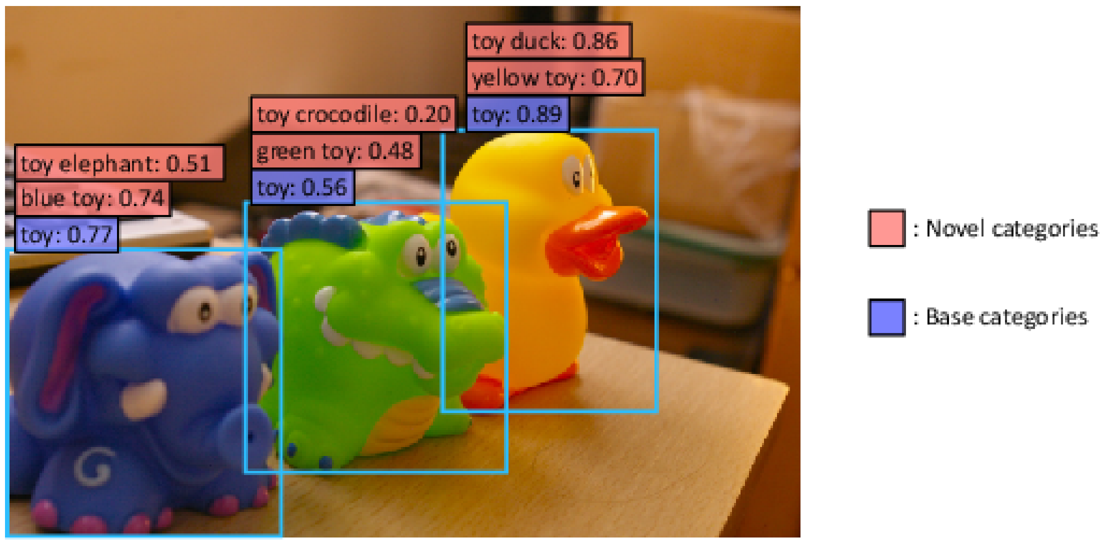


Figure 2.2: Illustration of an open-vocabulary object detection system in action, showcasing the ability to associate multiple descriptive captions with detected objects. The system accurately identifies various toys with different levels of specificity, from basic category labels such as `toy` to more detailed descriptions such as `toy elephant` and `toy duck`. The varied captions, with confidence scores, demonstrate the model's capability to recognize and describe objects beyond its base training categories. Adapted from *Open-vocabulary Object Detection via Vision and Language Knowledge Distillation*

The core innovation in open-vocabulary object detection lies in its use of **language embeddings** as a replacement for the final classification layer typically found in object detection architectures. Language embeddings are high-dimensional representations of words or phrases that capture semantic relationships between them. By integrating these embeddings, an open-vocabulary model can understand and detect a broader array of objects, including those not present in its training data. This approach relies on the semantic connection between the language representation of an object and the visual features extracted from images. For example, even if a model has never seen a "zebra" during training, it can infer its presence in an image by associating the visual features with the semantic context provided by language embeddings related to animals or patterns known to the model.

In practice, when an open-vocabulary object detection model encounters an image, it generates predictions for object locations and uses the integrated language embeddings to semantically interpret and label these objects, extending its detection capabilities to include categories beyond its training scope. This technique not only enhances the model's versatility and applicability to diverse real-world scenarios but also addresses the scalability issue of continually updating models with new object classes.

2.2 Robotics in Home Environments

The integration of robotics within home environments poses a unique set of challenges and opportunities, highlighting the necessity for comprehensive benchmarks and standardized platforms that facilitate innovation, evaluation, and comparison across different robotic systems.

Home environments are inherently complex and dynamic, featuring diverse layouts, obstacles, and a variety of objects that robots must navigate and manipulate. This complexity necessitates robust, versatile robotic systems capable of understanding and interacting with their surroundings in a meaningful way. However, benchmarking these systems has historically been difficult due to the variability in hardware used and the tasks performed. Many robotics challenges have allowed participants to use their own platforms, complicating fair comparisons of algorithms and systems [3]. Conversely, events providing a standardized robotic platform often changed tasks over time, making it challenging to track progress [16].

The pursuit of standardized robotics benchmarks has been a longstanding goal within the field. Efforts have included open-sourcing robot designs and introducing low-cost robots [17] to democratize access to cutting-edge research. However, the varied environments in which these robots operate often lead to the isolated evaluation of components—such as object navigation and Simultaneous Localization and Mapping (SLAM)—without considering their impact on the system as a whole. This gap highlights the need for end-to-end benchmarking platforms that evaluate individual components within the context of a fully integrated system.

Recognizing these challenges, recent initiatives have focused on creating reproducible benchmarks that support a wide range of tasks in complex, human-centric environments. The definition of Open-Vocabulary Mobile Manipulation (OVMM) as a key task for in-home robotics represents a significant step forward [18, 19, 20, 21, 22, 23, 24, 25]. This approach emphasizes the importance of developing and evaluating full-stack integrated mobile manipulation systems capable of operating in diverse settings with open object sets. By providing benchmarks and infrastructure both in simulation and the real world, researchers can build systems that are not only effective in controlled environments but also adaptable to the unpredictability of real-world applications.

2.3 OVMM: Open-Vocabulary Mobile Manipulation

The introduction of the OVMM benchmark marks the first reproducible mobile-manipulation benchmark designed for real-world applications, complemented by an associated simulation component. In simulation, the use of a dataset comprising 200 human-authored interactive 3D scenes [26], instantiated in the AI Habitat simulator [27, 28], creates a variety of challenging multi-room problems with a broad range of objects. This setup tests the robots' abilities to navigate and manipulate objects they have seen during training and those they have not. Similarly, in the real world, a controlled apartment environment serves as the benchmarking space, featuring a mix of seen and unseen object categories. The use of the Hello Robot Stretch [29] — a compliant and affordable robot — underscores the benchmark's commitment to accessibility and practical application in home environments.

This comprehensive approach to benchmarking in robotics research underscores the importance of creating standardized, reproducible platforms that facilitate the development, evaluation, and comparison of robotic systems.

2.4 HomeRobot 3D Simulation Environment

HomeRobot [30] is an affordable robot designed for navigating homes and manipulating objects to perform daily tasks. The **HomeRobot software framework** supports benchmarking in simulated and physical environments, featuring **identical APIs** across these settings. This allows for experiments to be consistently replicated, facilitating a direct comparison between simulated and real-world application outcomes.

The significance of employing simulation in the training and testing of embodied AI agents stems from several limitations associated with real-world interactions.

- Real-world training is inherently slow, as it cannot exceed or be parallelized beyond real-time speeds.
- It also poses potential dangers, where inadequately trained agents might cause harm to themselves, humans, or their environment.

- Moreover, the costs associated with physical agents and environments are substantial, and controlling or reproducing specific conditions or experiments in the real world is challenging.

In contrast, simulations offer a safer, cost-effective, and controllable environment that can operate significantly faster than real-time and be scaled across computing clusters. This approach not only ensures the safety and feasibility of developing and testing AI agents but also allows for systematic and fair benchmarking of progress. Once validated in simulation, these advancements can then be adapted to physical platforms.

Habitat-Lab, a component of AI Habitat, is a high-level library designed for end-to-end development in Embodied AI. It encompasses the definition of AI tasks, configuration of agents, training methodologies, and benchmarking of agent performance across standardized metrics. Habitat-Lab facilitates research by providing a comprehensive framework for exploring various aspects of Embodied AI, from navigation and interaction to instruction following and question answering.

HomeRobot is a framework developed on top of AI Habitat, inheriting and significantly extending its capabilities, with a focus on applications tailored for domestic environments. This sophisticated framework is designed to enable the development of robots capable of performing everyday tasks in these settings, achieving high levels of autonomy and adaptability. A key component of the HomeRobot simulation environment is its use of a subset of the Habitat Synthetic Scenes Dataset (HSSD), as described in Section 2.4.6, which offers a rich collection of synthetic scenes designed to mirror real-world domestic spaces closely. The incorporation of HSSD not only enriches the HomeRobot framework with a broad spectrum of scenarios and objects for simulation but also enhances the realism and relevance of the simulated tasks, ensuring the robots developed within this framework are well-equipped for real-world applications in domestic settings.

Figure 2.3: AI Habitat Simulation Environment.

Figure 2.4: Habitat Synthetic Scenes Dataset (HSSD) in the HomeRobot 3D Simulation Environment.

2.4.1 HomeRobot Key Features

Overall, the HomeRobot framework incorporates several key features:

- **Transferability:** A fundamental aspect of HomeRobot is its unified state and action spaces across simulation and real-world environments for each task. This facilitates an effortless transition between testing modes, allowing for control over the robot using either high-level action spaces, such as pre-defined grasping policies, or through low-level continuous joint control for detailed manipulation tasks.
- **Modularity:** HomeRobot's architecture supports modularity in perception and action components. This includes the incorporation of high-level states, such as semantic maps or segmented point clouds, and high-level actions, such as moving to a specified goal position or executing object pickup tasks. This modularity enhances the robot's interaction capabilities with its environment and simplifies the integration and testing of new features.
- **Baseline Agents:** Included within HomeRobot are baseline agents that utilize the stack's capabilities to offer basic functionalities necessary for Open Vocabluary Mobile Manipulation (OVMM) tasks. These agents are equipped with policies that facilitate fundamental operations, including navigation to particular locations and object manipulation.

2.4.2 OVMM Task

The **Open-Vocabulary Mobile Manipulation (OVMM)** task focuses on executing precise movement instructions for small, manipulable household items within a domestic environment. The task instructions are centered and emphasized as follows:

```
"Move (object) from the (start_receptacle) to the
                  (goal_receptacle)."
```

Here, the `object` represents any small household item such as a cup, stuffed toy, or box, while the `start_receptacle` and `goal_receptacle` refer to larger pieces of furniture that can support these items.

The primary challenge for the robot is to function within an unknown single-floor home environment, such an apartment, successfully identifying and manipulating the specified object based on the labels of `start_receptacle`, `object`, and `goal_receptacle`. The initial placement of the object on the `start_receptacle` acts as a directive for the robot to initiate its retrieval and relocation operation.

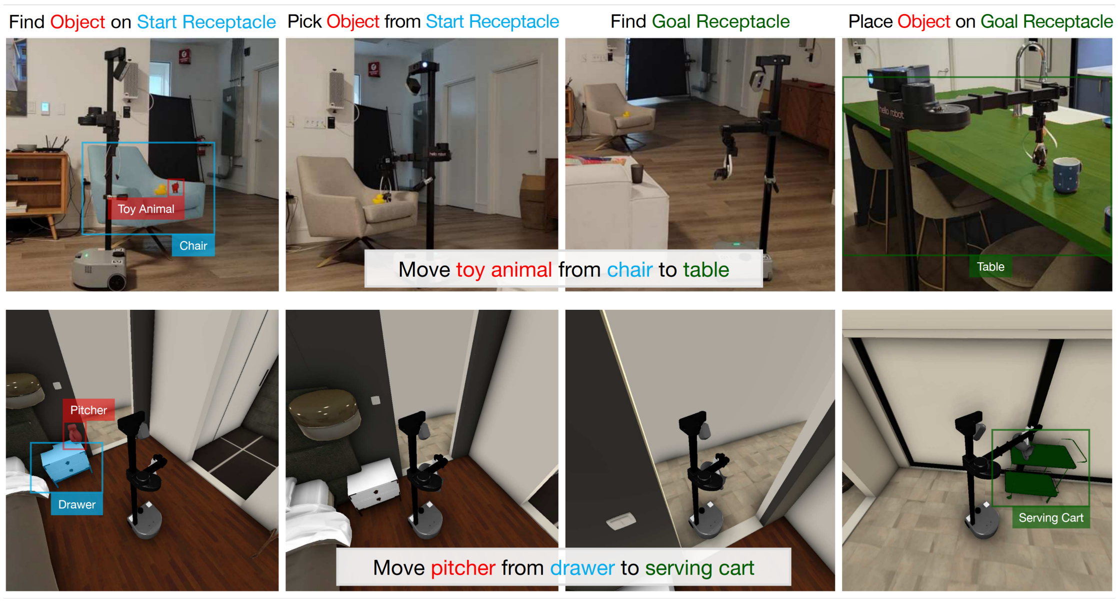

Find Object on Start Receptacle
Pick Object from Start Receptacle
Find Goal Receptacle
Place Object on Goal Receptacle
Toy Animal
Chair
Move toy animal from chair to table
Table
Pitcher
Drawer
Move pitcher from drawer to serving cart
Serving Cart

Figure 2.5: Sequential stages of the Open-Vocabulary Mobile Manipulation (OVMM) task performed in HomeRobot. **Top row**: The robot locates a toy animal on a chair (`start_receptacle`), grasps it, identifies the table (`goal_receptacle`), and places the object upon it. **Bottom row**: The robot finds a pitcher within a drawer, retrieves it, then moves to the serving cart (`goal_receptacle`), where it completes the task by placing the pitcher. These images illustrate the task's components: object detection, manipulation, and goal-oriented navigation within a home environment. Adapted from *HomeRobot: Open-Vocabulary Mobile Manipulation* [30].

Therefore, each episode is defined by the three elements: `start_receptacle`, `object`, and `goal_receptacle`. The agent is successful in an episode if the specified `object` is indeed moved from a `start_receptacle` on which it began the episode, to any valid `goal_receptacle`. The evaluation process awards points for each step of the task:

- **`FindObj/FindRec`:** Locate an object on a `start_receptacle`; or find a `goal_receptacle`.
- **`Gaze`:** Move close enough to an object to grasp it, and orient head to get a good view of the object. The goal of the gaze action is to improve the success rate of grasping.
- **`Grasp`:** Pick up the object. The framework provides a high-level action for this, since they do not simulate the gripper interaction in Habitat.
- **`Place`:** Move to a location in the environment and place the object on top of the `goal_receptacle`.

In the context of the Open-Vocabulary Mobile Manipulation (OVMM) challenge, the agent is designed to recognize and interact with a predefined set of 150 classes. These classes include a variety of receptacles and objects, highlighted in Appendix A.1.

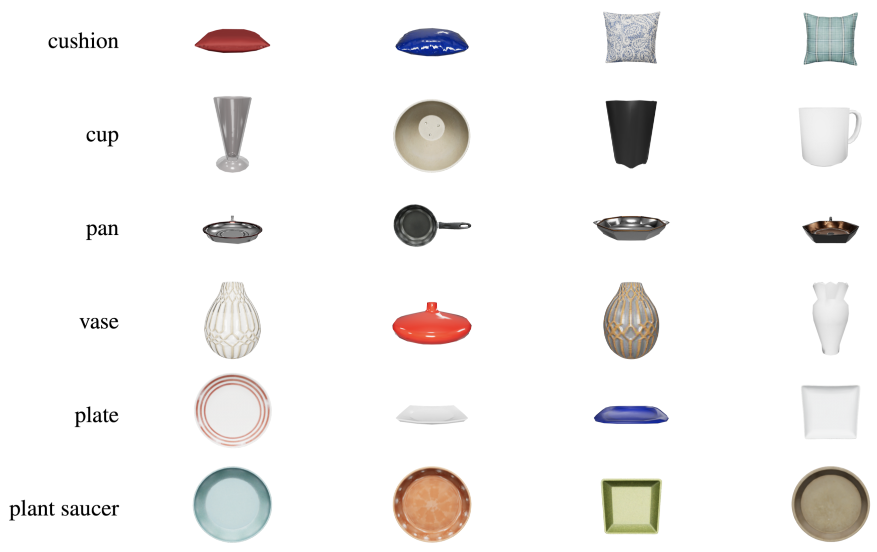


Figure 2.6: Examples of object models involved in the OVMM tasks. Adapted from *HomeRobot: Open-Vocabulary Mobile Manipulation* [30].

2.4.3 HomeRobot Structure

The structure of the HomeRobot framework is a direct reflection of its key features. The HomeRobot library is organized into three distinct repositories, each serving a specific purpose:

- `home_robot`: This repository houses shared components essential for the framework's operation, including `Environment` interfaces, controllers, and modules for detection and segmentation. It forms the core of HomeRobot, where most policies are implemented, reflecting the framework's emphasis on modularity.
- `home_robot_sim`: Dedicated to the simulation stack, this repository builds on the capabilities of Habitat to provide `Environments` tailored for simulation. It underscores the framework's transferability feature by ensuring that the simulated environments mimic real-world conditions as closely as possible.
- `home_robot_hw`: Focused on the hardware stack, it includes server processes that run directly on the robot, a client API designed for the GPU workstation, and `Environments` constructed using this client API. This setup facilitates real-world applications of the HomeRobot, allowing for direct control and interaction with the physical robot.

Within HomeRobot, functionality is divided between `Agents` and `Environments`, adopting a structure common to many reinforcement learning benchmarks. This separation enhances the framework's modularity:

- `Agents` are responsible for executing policies. HomeRobot implements agents that utilize a combination of heuristic policies and those learned through Reinforcement Learning on scene datasets. This dual approach enables the framework to support a wide range of tasks and adapt to various challenges encountered in robotic applications.
- `Environments` provide the logical framework that interacts with Agents. They supply `Observations` to the `Agents` and offer a function to apply the Agent's actions within the environment, whether real or simulated. This design allows for a clear delineation of responsibilities, simplifying the development and testing of new functionalities within the HomeRobot framework.

2.4.4 Open Vocabulary Manipulation Agent

The HomeRobot framework outlines two primary baseline approaches: the heuristic baseline, which relies on established motion planning techniques and simple rules for grasping and manipulation, and a reinforcement learning baseline, leveraging the DD-PPO algorithm for policy learning. In this study, we focus on the **heuristic baseline** within the HomeRobot framework, aiming to enhance its capabilities for executing Open-Vocabulary Mobile Manipulation (OVMM) tasks with the Stretch robot. Set in both simulated and real indoor environments, the challenge involves moving an object from a start to an end receptacle, both of which are classes of furniture that may be present in multiples places within the environment. For instance, if the end receptacle is a "table", any table within the environment qualifies as a correct destination.

The heuristic baseline agent is designed with a structured skill set, encompassing detection, exploration, navigation, picking, and placing:

1. **Detection** involves object detection and segmentation using the agent's camera RGBD images and constructing a Bird's Eye View (BEV) map to identify semantic areas within the environment.

2. **Exploration** entails searching the environment to locate the start and end receptacles, and exploring the vicinity of start receptacles to find the target object.

3. **Navigation** directs the agent towards a specified goal, which could be the object itself or the start/end receptacle, facilitating movement within the environment.

4. **Picking** comprises lifting the object, supported by a high-level action command that abstracts the complexities of simulated gripper interaction, a feature not present in Habitat. This skill is enabled by multiple grasping strategies and policies for learning optimal grasping techniques.

5. **Placing** adjusts the agent's position to accurately place the object at the end receptacle, completing the task.

The execution of these skills follows a specific sequence, starting with a 360° turn by the agent to survey its immediate surroundings. Utilizing the Exploration skill, the agent seeks the object, moving towards it upon discovery and employing the Picking skill to secure it. If the end receptacle is not immediately found, the agent re-engages in exploration; otherwise, it navigates towards the nearest end receptacle, concluding the task with the Placing skill. Throughout its operation, the agent continuously applies the Detection skill to gather information on potential start and end receptacles, even when the primary goal is object retrieval.

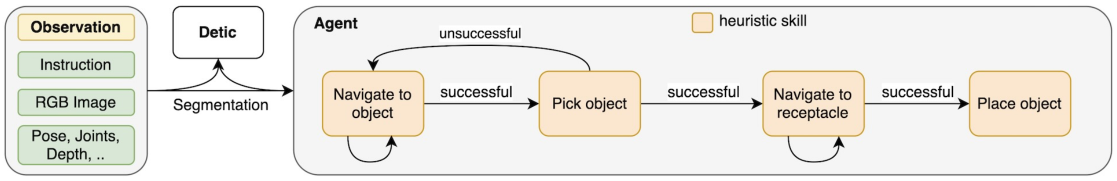


Figure 2.7: The architecture of our agent. It is build as a state machine with six states: Find object, Navigate to object, Pick object, Find end receptacle, Navigate to end receptacle, and Place object. In the Find-phases, the Exploration skill is used. The agent's perception model is Detic, it segments the object and receptacles in the RGB image. This segmentation, together with other observations such as the pose, joints and a depth image, are the input to the agent. Adapted from *UniTeam: Open Vocabulary Mobile Manipulation Challenge* [31].

The perception capabilities are powered by the **Detic** model, which generates masks for objects and receptacles, enabling the agent's understanding of its environment and influencing its actions.

2.4.5 3D Scene Datasets

Recent advancements in embodied AI have been significantly supported by the development and use of 3D scene datasets, which are essential for training AI agents to navigate realistic environments, follow language instructions [32], find and rearrange objects [33, 34, 28, 35], among other tasks.

These datasets have become the cornerstone of simulation platforms [2, 36, 37, 38], facilitating safe, systematic, and scalable training and evaluation of AI agents. 3D reconstructions and synthetic datasets, with arrangements of human-designed objects, have been central to mimicking real-world complexity and diversity [39, 40, 41, 42]. However, reconstruction datasets often come with challenges such as noise, missing geometry, and other artifacts that could potentially skew the training process and lead to overfitting. The labor-intensive nature and difficulty in scaling the acquisition and annotation of these reconstructions, combined with their rigid, non-manipulable nature, limit their usefulness in scenarios requiring environmental interaction [33]. As a result, there has been a shift towards **synthetic 3D scenes**, constructed from human-authored objects, to better represent real-world environments [43, 44, 28]. Despite their widespread use, there's been little systematic analysis comparing the scale and realism of these datasets, or how these factors impact the generalizability and performance of AI agents. Additionally, while procedural scene generation [43] offers potentially limitless scaling of datasets, the benefits of this scale on task performance remain underexplored.

2.4.6 HSSD: Habitat Scene Synthesis Dataset

The **Habitat Synthetic Scenes Dataset** (HSSD-200) marks a significant contribution towards achieving realism in synthetic environments for embodied AI research. This dataset comprises recreations of real residential spaces, constructed using a diverse array of 18,656 unique and high-quality 3D models of real-world objects. Set to be open-sourced and freely available under an academic research license, HSSD-200 offers an unprecedented level of visual fidelity, accurate scene dimensions, and realistic object occurrence statistics compared to previous synthetic datasets. Through systematic evaluation, including a scale versus realism study, HSSD-200 has demonstrated its superiority in training **ObjectGoal navigation** agents. These agents, tasked with navigating towards specific objects (e.g., bed, tv, chair), exhibit enhanced performance and generalization to real-world scenes, such as those from HM3DSem [40] and MP3D [39], even when trained on a considerably smaller number of scenes from HSSD-200. Remarkably, training on merely 122 scenes from HSSD-200 results in better generalization than training on 10,000 scenes from ProcTHOR [43]. Beyond navigation, the compositional nature of HSSD-200 facilitates research in object manipulation and rearrangement tasks [28], offering flexibility for adding or removing objects.

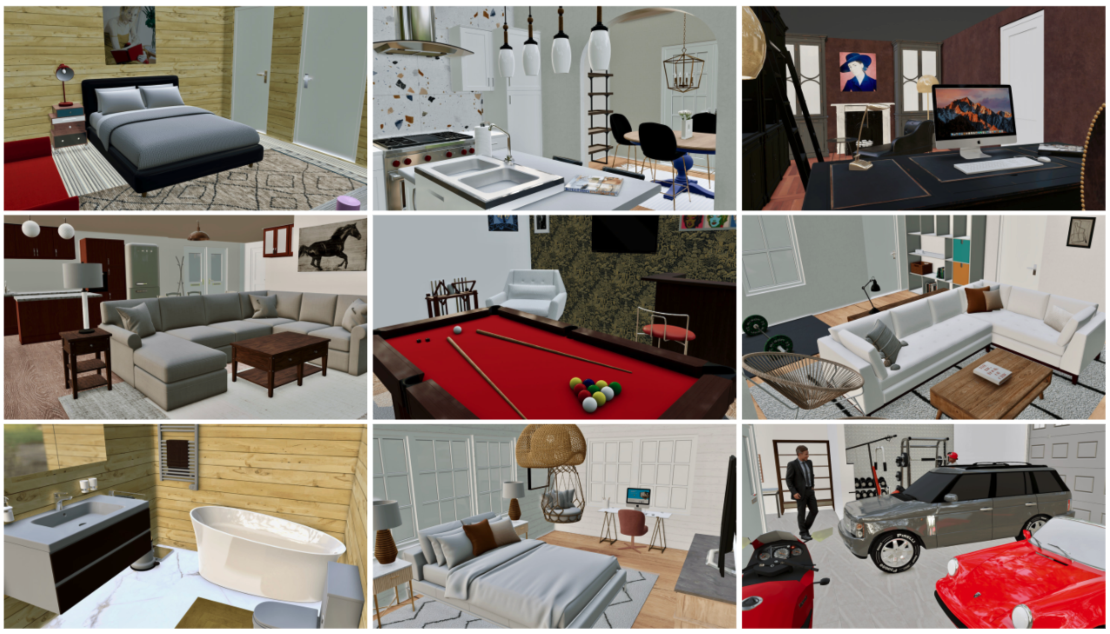


Figure 2.8: Examples of scenes from the Habitat Synthetic Scenes Dataset (HSSD).

The **HomeRobot simulation environment** [30] is built on a subset of the

Habitat Synthetic Scenes Dataset (HSSD), incorporating high-quality 3D scenes that enhance its functionality and realism. This integration allows HomeRobot to provide a more realistic and engaging experience for testing and developing embodied AI agents, facilitating research in navigation, object manipulation, and other tasks requiring interaction with an environment that closely mirrors the real world. The use of HSSD's meticulously designed scenes ensures that agents trained within the HomeRobot simulation can benefit from the dataset's attention to detail in visual fidelity, spatial accuracy, and object diversity, promising better preparation for real-world application.

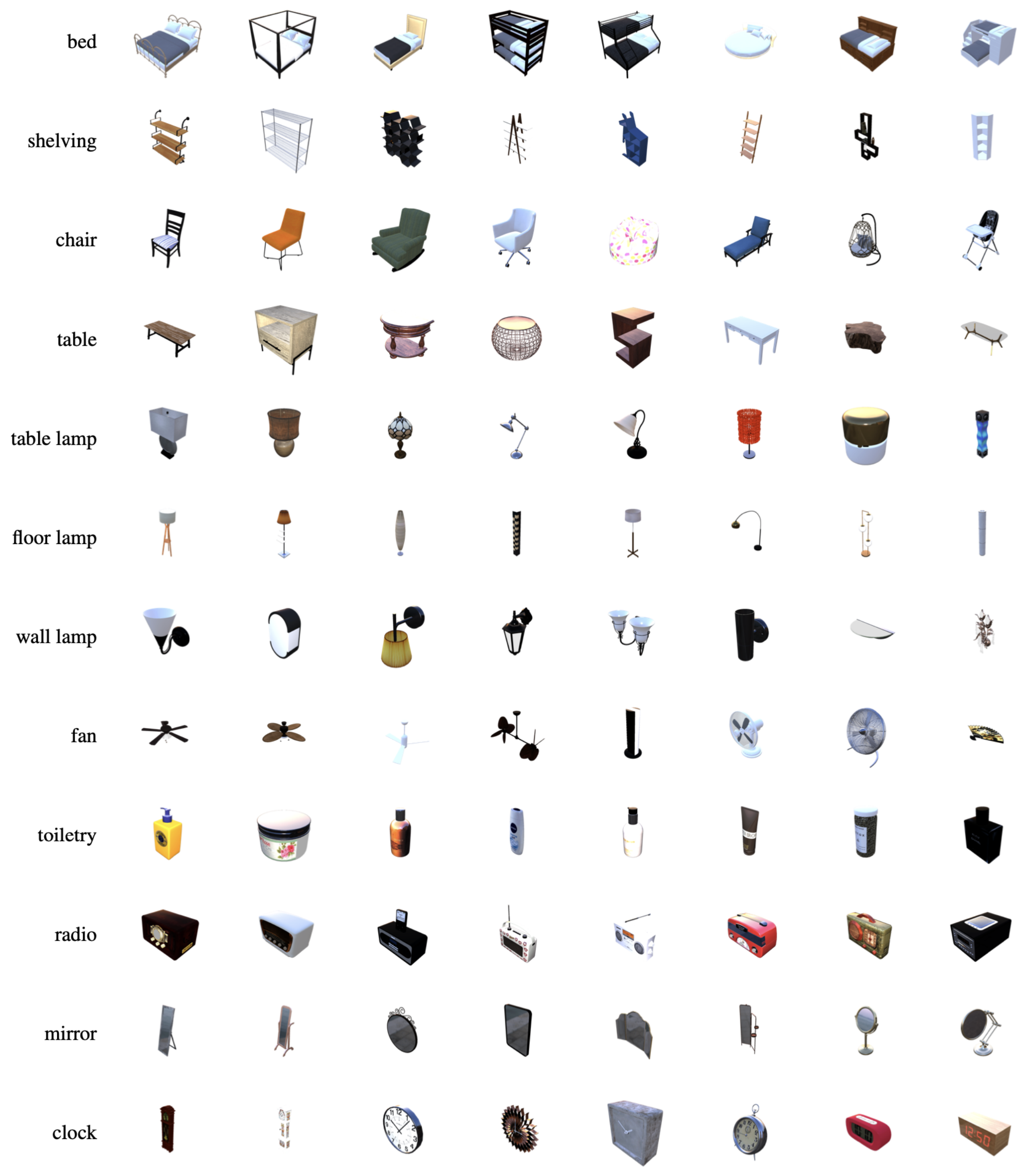


Figure 2.9: Examples of object models involved in the OVMM tasks. Adapted from *HomeRobot: Open-Vocabulary Mobile Manipulation* [30].

2.4.7 Episode Generation

Episode generation in the Open-Vocabulary Mobile Manipulation (OVMM) framework starts with selecting scenes from the Habitat Synthetic Scenes Dataset (HSSD). The first step in generating an episode is to identify the largest navigable indoor area in each scene to define the agent's operational space. After determining this area, the next step involves filtering out receptacles within the scene that are too small for object placement, ensuring only viable receptacles are used for start and goal locations. An object is then placed on one of these filtered start receptacles, and a goal receptacle is chosen for the object's intended placement. The agent is finally spawned at a random point within the navigable area of the house to start the episode. This process ensures that the episodes are set up in a way that maximizes the variety available for navigation and manipulation.

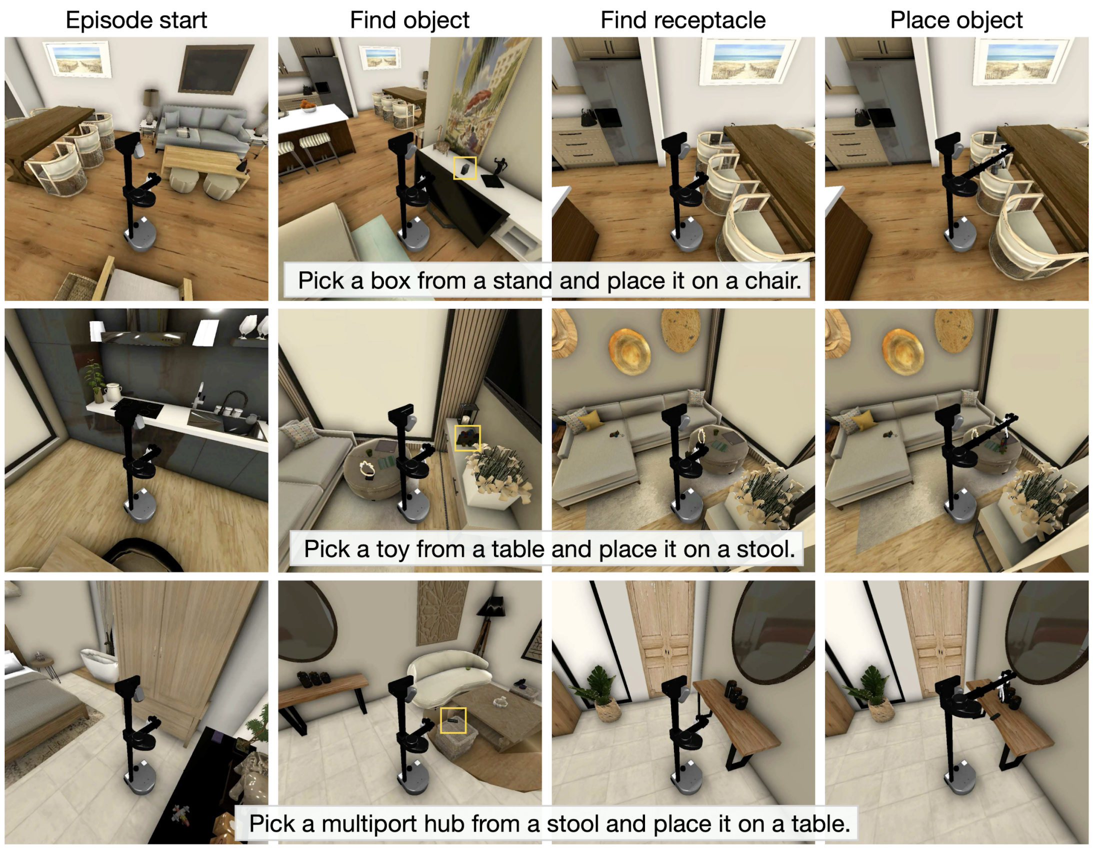

Episode start
Find object
Find receptacle
Place object
Pick a box from a stand and place it on a chair.
Pick a toy from a table and place it on a stool.
Pick a multiport hub from a stool and place it on a table.

Figure 2.10: Examples of OVMM task episodes. The top row demonstrates the sequence of an episode: from the initial spawn point of the agent ('Episode start'), through 'Find object', to 'Find receptacle', and finally 'Place object'.

2.5 DETIC: Detecting Twenty-thousand Classes using Image-level Supervision

DETIC, standing for **Detector with Image Classes**, marks a pioneering approach within the object detection domain by addressing its two principal challenges:

1. **Localization**: The identification of objects' presence and location within an image.
2. **Classification**: Assigning accurate labels to each detected object.

Unlike conventional methods that merge these tasks—requiring datasets with bounding box annotations for all classes—DETIC recognizes a significant gap in dataset capacities. For instance:

- **LVIS** detection dataset [45] offers more than 1,000 classes across 120,000 images.
- **OpenImages** [46] encompasses 500 classes within 1.8 million images.
- Contrastingly, **ImageNet** [47] for image classification, features 21,000 classes from 14 million images.

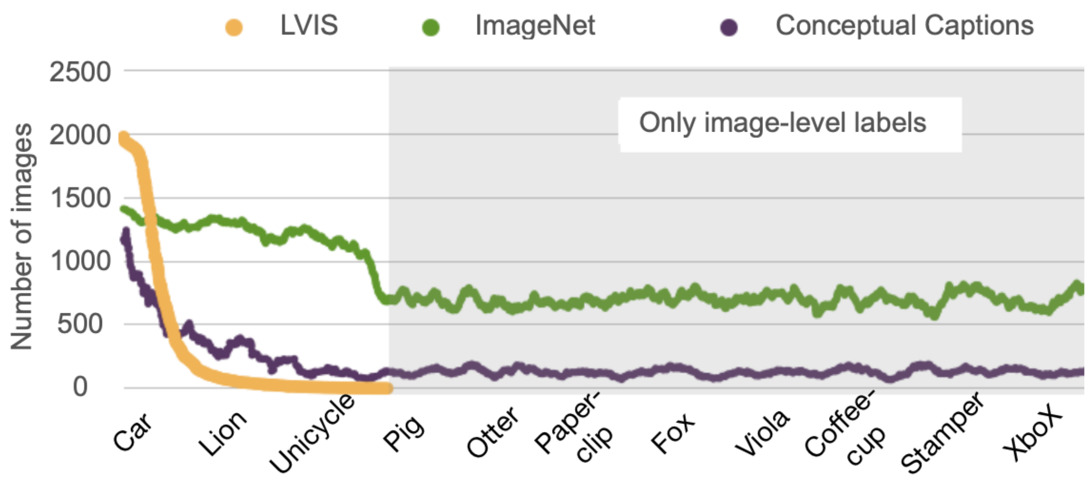


Figure 2.11: Number of images in LVIS, ImageNet, and Conceptual Captions per class (smoothed by averaging 100 neighboring classes). Classification datasets have a much larger vocabulary than detection datasets. Adapted from *Detecting Twenty-thousand Classes using Image-level Supervision* [48]

This disparity, especially the lack of extensive annotations for many classes, poses a challenge for training robust object detectors.

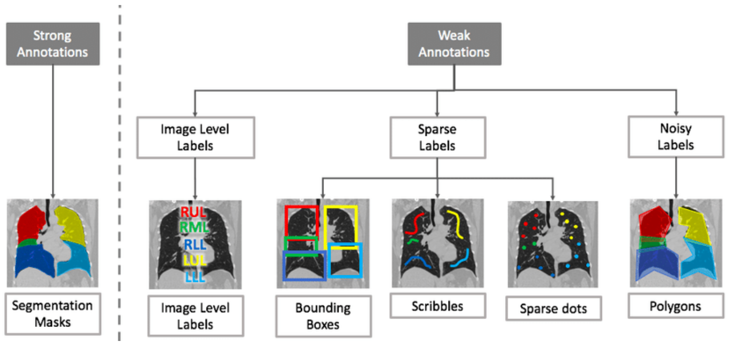
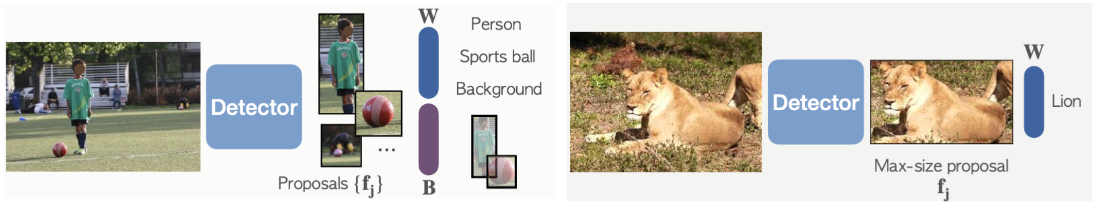


Figure 2.12: An example of strong and weak annotations in object detection. Adapted from «Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation»

DETIC seeks to bridge this gap by utilizing **image-level supervision** alongside traditional detection training, expanding the model's vocabulary and its object classification capability. It decouples the localization and classification tasks, capitalizing on modern region proposal networks' ability to localize objects without direct supervision. Specifically, DETIC:

- Focuses on the classification challenge by employing image-level labels to train the classifier.
- Implements a straightforward classification loss targeting the **largest proposal** within the image, without supervising other outputs for image-labeled data.



(a) Detection data **(b)** Image-labeled data

Figure 2.13: Approach Overview: The model is trained using both detection data and image-labeled data. With detection data, standard detection losses are applied to train both the classifier (W) and the box prediction branch (B) of the detector. In contrast, when image-labeled data is utilized, training is exclusively focused on the classifier through a modified classification loss. This loss targets the features extracted from the **largest-sized** proposal within the image. Adapted from *Detecting Twenty-thousand Classes using Image-level Supervision* [48]

This method significantly broadens the detection vocabulary and simplifies the model's training process.

In contrast to existing weakly-supervised detection methodologies [50, 51, 52, 53] that navigate both localization and classification with weakly labeled data, DETIC innovates by solely concentrating on the classification aspect when leveraging classification data. This avoids the complex prediction-based label assignment process—a circular challenge where accurate label assignment depends on precise detections, which in turn requires extensive labeled data for training.

The DETIC model enhances its detection capabilities by employing **CLIP embeddings** as classification weights, a strategic move that significantly broadens its ability to perform open-vocabulary object detection. This approach enables DETIC to:

- Identify objects beyond its initial training set, addressing the challenge of **zero-shot object detection**.
- Offer a more nuanced understanding of class names and expand its detection range, advancing beyond models that utilized other language embeddings.

DETIC integrates **CLIP embeddings**, distinguishing itself from previous models such as OVR-CNN [54], ViLD [15], OpenSeg [55], langSeg [56]. These models also utilize CLIP embeddings to improve their language models, but DETIC sets itself apart by:

- Directly applying CLIP embeddings as classification weights without relying on feature distillation from CLIP's image features.
- Employing a co-training method with additional image-labeled data, thereby simplifying the model architecture and enriching the training dataset to enhance its object classification capabilities.

2.5.1 DETIC Model Overview

The DETIC model utilizes the **Swin Transformer** [57] as its foundational structure for image processing. This model processes an RGB image frame by initially breaking it down into smaller patches through a process known as **patch partitioning**. These patches are then linearly encoded and passed through the Swin Transformer blocks, which are arranged in stages that build upon each other.

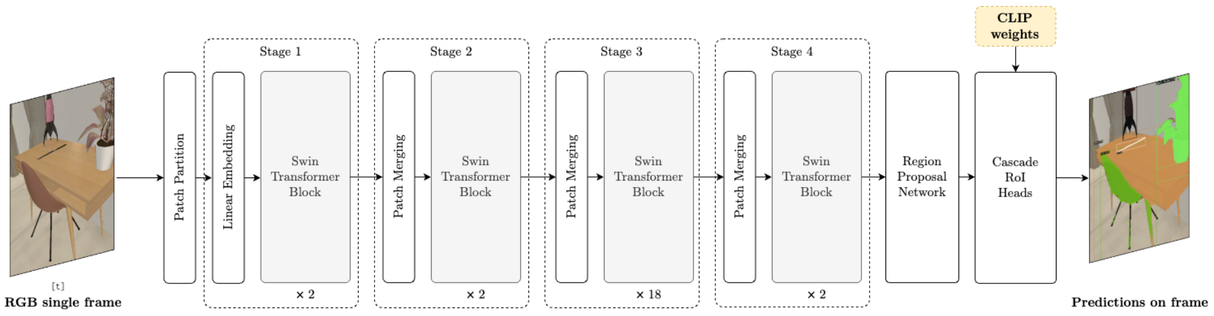


Figure 2.14: The DETIC model architecture, featuring the Swin Transformer backbone and the Region Proposal Network (RPN) for object detection.

The first two stages consist of few Transformer blocks (denoted by ‘x2‘), which serve to extract preliminary features from the image. These stages capture basic visual elements and set the stage for more complex pattern recognition. The third stage, however, significantly increases in complexity (indicated by ‘x18‘) and is dedicated to deeply analyzing the image data to identify intricate details that are crucial for accurate object recognition.

Swin Transformer’s approach involves calculating self-attention within **local windows** which are shifted at each layer to cover different parts of the image. This strategy allows the DETIC model to maintain **computational** efficiency while scaling to handle larger images and more complex patterns. It’s this hierarchical and overlapping window-based self-attention mechanism that allows the Swin Transformer to capture a wide variety of visual features, from the most granular details to broader structural patterns, which is particularly advantageous for image classification tasks that require understanding both the parts and the whole.

After the Swin Transformer refines the image features, the DETIC model employs Region Proposal Network (RPN) to generate candidate object bounding boxes. These proposals are refined through a **Region of Interest** (RoI) pooler, which resizes the features to a fixed size, making them ready for classification. Within the RoI heads, DETIC employs **CLIP weights** to classify a broad array of object classes effectively, using zero-shot learning capabilities. The final step involves DETIC producing a set of scores for each object class and refining the bounding box coordinates for localization.

In conclusion, the DETIC model is composed of three key components:

- A **backbone** that leverages the Swin Transformer, providing a powerful architectural foundation for feature extraction.
- A **Heatmap based Proposal Generator** utilizing CenterNet2, which efficiently generates object proposals within the scene.
- **Cascade RoI Heads** that refine the object proposals through a cascaded process, leading to precise object detection and segmentation. The classification stage of the RoI heads is enhanced by the integration of CLIP weights, enabling zero-shot object detection capabilities.

In our further discussions and enhancements, we will primarily focus on the *backbone* with the Swin Transformer and the *Cascade RoI Heads* with the CLIP weights for classification.

2.5.2 Swin Transformer Backbone

The Swin Transformer represents a novel adaptation of the Transformer model, traditionally utilized for language processing tasks, to the domain of vision, introducing a **hierarchical vision transformer architecture**. This approach addresses the unique challenges of visual data, including the varying scale of visual entities and the high resolution of images, by constructing a hierarchical representation with shifted windowing schemes. Such a design significantly enhances computational efficiency by limiting self-attention computation to non-overlapping local windows, while simultaneously enabling cross-window connections through a shifting strategy. As a result, the Swin Transformer **scales linearly** with the image size, demonstrating its suitability for a broad array of vision tasks, ranging from image classification to dense prediction tasks such as object detection and semantic segmentation.

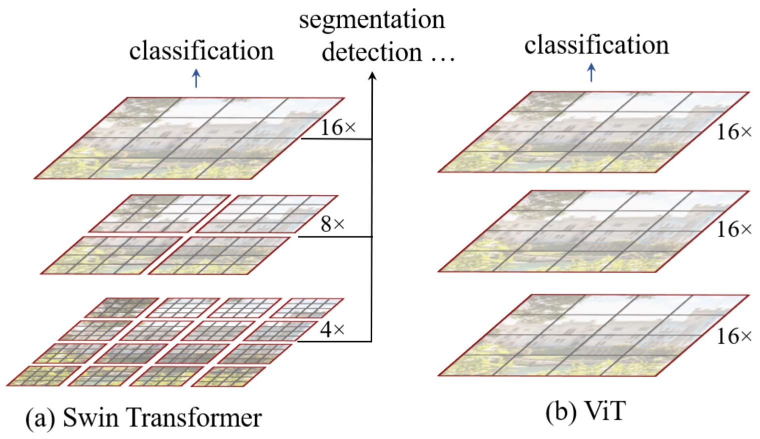


Figure 2.15: (a) The Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally. Adapted from *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows* [57].

The hierarchical architecture begins by dividing the input image into **fixed-size patches**, treating each patch as a "token" in the initial layer, akin to words in a text for language models. This structure starts with small-sized patches and gradually merges them in deeper layers, maintaining a hierarchy that supports modeling at various scales and complexities. The core of the Swin Transformer block is its unique approach to self-attention, employing a shifted window-based self-attention (SW-MSA) mechanism.

The model alternates between two window partitioning configurations across consecutive layers to compute self-attention. Initially, it employs regular window partitioning, followed by a diagonal shift in the next layer. This alternating scheme ensures efficient computation and introduces connections across windows, significantly enhancing the model's ability to capture relationships between different parts of the image. By restricting self-attention to local windows and leveraging a hierarchical structure, the Swin Transformer achieves a computational complexity that scales **linearly** with the image size. This represents a critical improvement over traditional Transformer models that exhibit **quadratic complexity**. The hierarchical nature of the Swin Transformer, combined with its efficient self-attention mechanism, makes it adaptable to a wide spectrum of vision tasks, achieving state-of-the-art results on benchmarks for image classification, object detection, and semantic segmentation.

The process begins with initial patch partitioning, where the input image is divided into fixed-size patches (e.g., 4x4 pixels). Each patch is then flattened and passed through a linear embedding layer to transform its raw pixel RGB values into a higher-dimensional feature vector. Swin Transformer blocks process these tokens, with each block consisting of a shifted window-based multi-head self-attention mechanism and a multilayer perceptron (MLP). The model employs Layer Normalization (LN) before each attention and MLP module, and residual connections after each module. Patch merging layers play a crucial role in the network's depth, gradually merging adjacent patches (e.g., groups of 2x2 patches) into larger patches. This effectively reduces the resolution while increasing the feature dimension, simulating the effect of pooling layers in CNNs and facilitating a hierarchical representation.

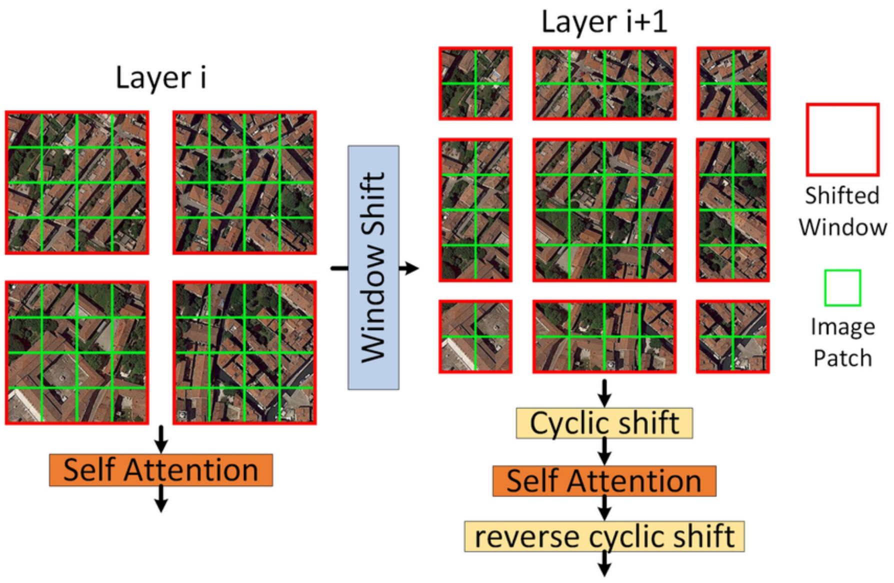


Figure 2.16: Shifted window strategy in Swin Transformer. Layer i demonstrates self-attention within local windows. Layer $i+1$ applies a window shift, enabling cross-window connectivity. A cyclic shift precedes self-attention, followed by a reverse to maintain relative positions. Adapted from «Meta-TR: Meta-Attention Spatial Compressive Imaging Network With Swin Transformer» [58]

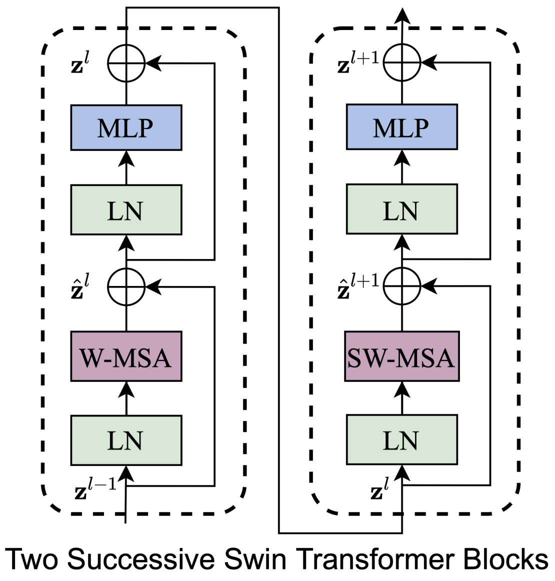


Figure 2.17: Two successive Swin Transformer Blocks. Adapted from *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows* [57].

2.5.3 Region of Interest (RoI) Heads with CLIP weights

Open-vocabulary object detection extends the capabilities of traditional object detection models, allowing for the recognition of objects outside the limited scope of the training dataset's predefined classes. The problem is twofold:

- **Localization**: Identifying all objects in an image $I \in \mathbb{R}^{3\times h\times w}$ and their locations, represented by bounding boxes $b_j \in \mathbb{R}^4$.
- **Classification**: Assigning a class label $c_j \in C_{\text{test}}$ to each detected object, where C_{test} is the class vocabulary defined at test time by the user.

Object detection systems utilize various class vocabularies during training and testing:

- C_{test}: Vocabulary of classes for testing.
- C_{det}: Vocabulary of classes used during the training of the detector.

The approach to handling these vocabularies differs between closed and open vocabulary detection:

- **Closed Vocabulary Detection**: The test and detection class vocabularies are the same ($C_{\text{test}} = C_{\text{det}}$).
- **Open Vocabulary Detection**: The test class vocabulary can include classes not present in the detection training ($C_{\text{test}} \neq C_{\text{det}}$).

An object detection model often follows a two-stage approach. First, a Region Proposal Network (RPN) generates initial object proposals. Then, the RoI Heads, central to the **classification stage**, refine these proposals. The RoI Heads receive pooled features for each proposed region and perform classification and regression tasks. Utilizing **CLIP weights** in the DETIC model, the RoI Heads facilitate zero-shot learning by aligning refined image features with textual descriptions through language embeddings, allowing the recognition of classes outside C_{det}, as shown in Figure 2.18.

Therefore, the innovation in open-vocabulary object detection comes with allowing $C_{\text{test}} \neq C_{\text{det}}$. Substituting the classification weights W with fixed language embeddings transforms a standard detector into an **open-vocabulary detector**. The model is trained so that region features align with these language embeddings.

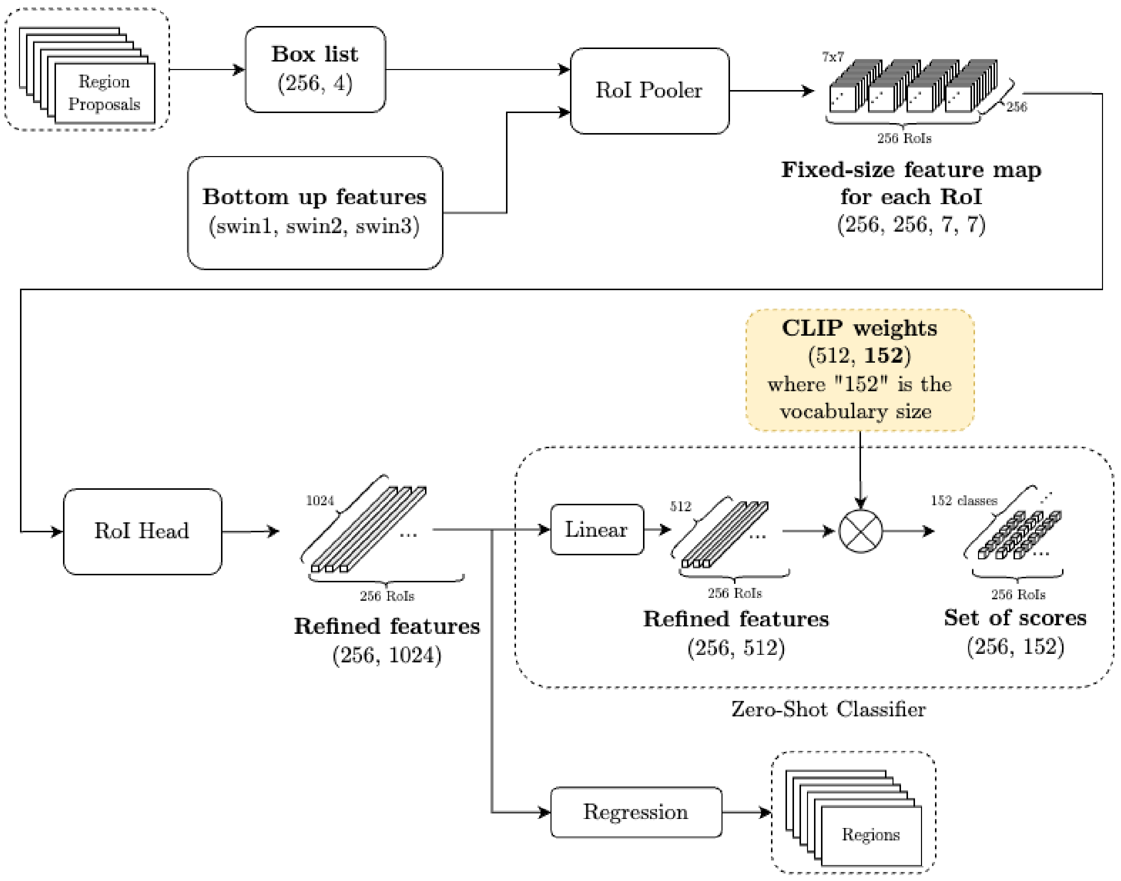


Figure 2.18: One stage of Region of Interest (RoI) Heads. Integration of **CLIP weights** in DETIC's RoI classification stage, enhancing zero-shot classification capabilities. Refined features from the RoI Head are mapped to class scores using CLIP embeddings.

The **DETIC** model, which was considered in this work, adopts the language embeddings from **CLIP** as classification weights following [15]. This model theoretically enables detection across any class, though practical applications may show varying results. DETIC addresses this by enhancing region features to better match the language embeddings, improving its ability to identify a diverse array of objects at test time as specified by C_{test}.

During the model preparation phase, an essential step involves the integration of CLIP embeddings for the 150 classes pertinent to the challenge. These classes coincide with those found in the dataset compiled for training.

A distinctive aspect of this process is the appending of a prompt, specifically the article **"a"**, to each class name before generating the embeddings. This procedure is not arbitrary; it is rooted in the architecture and training methodology of CLIP (Contrastive Language–Image Pre-training).

CLIP has been trained on a vast corpus of images and their corresponding textual descriptions, learning to associate the visual content with natural language annotations. By prepending a simple article such as **"a"** to the class names, we mimic the structure of the sentences found in the training data of CLIP, thereby aligning our input more closely with the model's learned representations.

This alignment is crucial for enhancing the model's ability to accurately generate embeddings that are semantically rich and contextually relevant to the given classes. These embeddings, once created, are utilized to reset the classifier of the target model, ensuring that the model is considering only the classes of the specific categories of interest in the challenge.

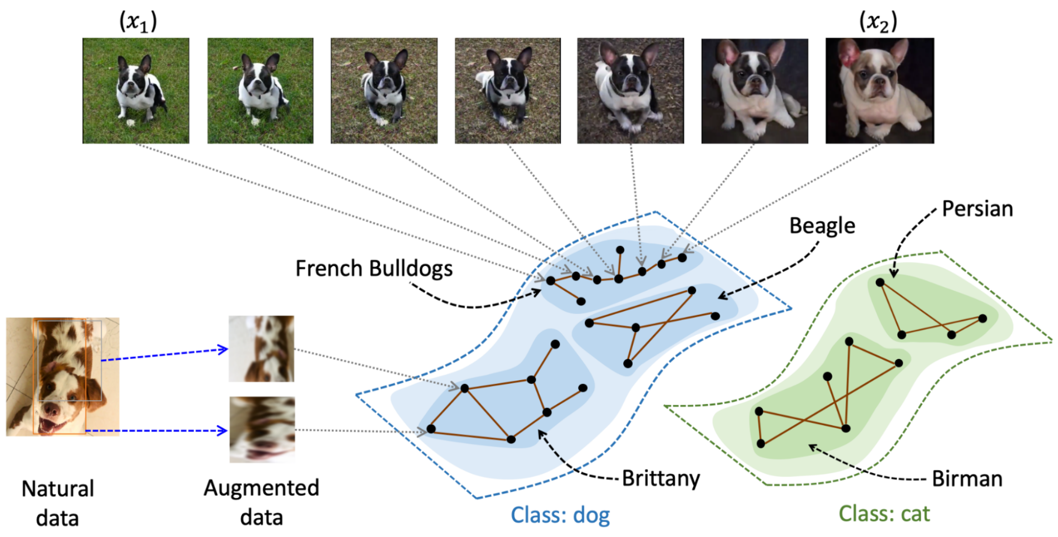


Figure 2.19: An example of contrastive learning in CLIP applied to distinguish between different classes of animals. The model is trained to pull closer the representations of augmented versions of the same image (e.g., different pictures of French Bulldogs) while pushing apart the representations of different classes (e.g., French Bulldogs versus Beagles or Persians). Adapted from 'Understanding Contrastive Learning' by the Stanford AI Lab (https://ai.stanford.edu/blog/understanding-contrastive-learning/)

2.6 Temporal Consistency in Object Detection

Temporal consistency in object detection is a critical aspect for robots operating in dynamic environments, such as those encountered in household navigation tasks. Achieving this consistency entails the object detection system's ability to recognize and track objects across consecutive frames within a video sequence, despite changes in object appearance, camera perspective, and environmental conditions. This becomes particularly challenging in egocentric vision, where the viewpoint shifts with the robot's or user's movements.

The concept of **Parameter-efficient Transfer Learning** has been pivotal in adapting knowledge from large pre-trained image-based models to the video domain with minimal additional parameters. In the NLP field, state-of-the-art performances across various tasks have been attained through such models, known as foundation models [59], like BERT [60] and GPT [61, 62]. Adaptation techniques range from complete fine-tuning to partial adjustments, such as linear probing. However, with the growing size and complexity of these foundation models, fully fine-tuning for each task becomes impractical due to exorbitant training costs and storage requirements [63, 64]. This poses significant challenges for their application in real-world settings.

This trend toward efficiency has sparked interest in the computer vision community. The success of the CLIP model [7], trained on a vast number of web image-text pairs, is testament to this. In the video domain, although the computational cost is substantially higher, the principles remain the same. Training video variants of such models can be restrictive due to the scarcity of large video datasets and the enormity of the required computing resources [65]. Hence, leveraging large pre-trained image models as the starting point for video tasks remains a favorable strategy.

2.6.1 Parameter-efficient Transfer Learning

Efficient adaptation of large pre-trained image models to video downstream tasks is a critical challenge, particularly in the context of action recognition. The need for efficiency arises from the significant differences in computational resources and training time required for video models compared to image models [66]. Bridging the gap between static images and dynamic videos during transfer learning is complex, as pre-trained image models lack the ability to decode temporal structures, which is essential for video interpretation.

State-of-the-art video models [67, 66, 68, 69] predominantly focus on incorporating the temporal aspect into existing image architectures. The process of adapting these models to video tasks often requires significant changes to the architecture, going well beyond simple fine-tuning. For each specific video task, models are typically subjected to thorough retraining or extensive fine-tuning. This is not just to fine-tune the model's spatial recognition abilities—which have been pre-learned from images—but also to develop its capacity to understand temporal sequences, a critical dimension in video understanding. The goal of such adaptations is to find an optimal balance: the models must retain the sophisticated spatial understanding obtained from pre-trained image models while also gaining the ability to interpret the temporal context that is essential for analyzing videos.

2.6.2 ST-Adapter: Spatio-Temporal Adapter

Incorporating dynamic temporal understanding into pre-trained image models, such as the Vision Transformer (ViT), is a critical step in adapting them to video-related tasks such as action recognition. State-of-the-art approaches typically involve integrating a temporal learning module and subsequently performing full network fine-tuning [70, 71]. However, this method is not parameter-efficient, as it produces a distinct, large-scale model for each task, demanding extensive computational resources.

Addressing this inefficiency, the **ST-Adapter** (Spatio-Temporal Adapter) [72] presents a more streamlined solution by introducing a spatio-temporal reasoning capability within a compact structure. The ST-Adapter allows an image model, which initially lacks temporal processing abilities, to analyze and understand video content. It does so with a minimal per-task parameter increase—less than 8%—and requires roughly 20 times fewer updated parameters compared to its full fine-tuning counterparts.

The pre-trained model weights remain **frozen**, ensuring the existing knowledge is retained, while only the weights of the adapter are updated during training. By focusing only on training this lightweight adapter, significant computational savings are achieved. This efficiency does not sacrifice performance; on the contrary, the ST-Adapter has been shown to match or even outperform full fine-tuning methods [73, 70, 66], offering an economical and effective approach to video task adaptation. The result is a highly parameter-efficient, cost-effective model that maintains, if not enhances, the accuracy for video understanding tasks.

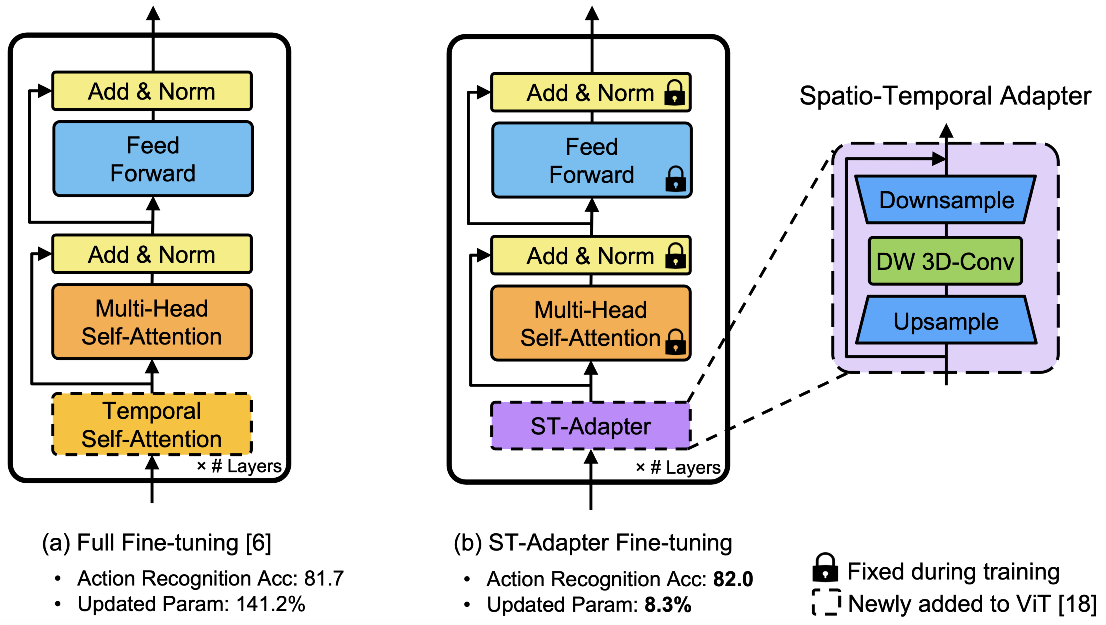


Figure 2.20: Architectural comparison between full fine-tuning and the incorporation of a Space-Time Adapter (ST-Adapter) into a Vision Transformer (ViT) for video tasks. (a) Traditional methods fully fine-tune the pre-trained image model, resulting in a high parameter cost for each downstream task. (b) The ST-Adapter approach introduces a compact module specifically designed for spatio-temporal reasoning, enabling efficient adaptation to video content with a minimal increase in parameters. Adapted from *ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning* [72]

Chapter 3

Methodology

3.1 Overview

This study focuses on enhancing temporal consistency in **egocentric object detection** for robotic navigation and interaction in dynamic home environments. Within the context of the **Open-Vocabulary Mobile Manipulation (OVMM)** challenge, our research is situated in the **HomeRobot 3D simulation environment**, utilizing a simulated Hello Robot Stretch for object manipulation tasks. The core of our methodology leverages open-vocabulary object detection models, specifically **DETIC**, which is capable of recognizing a broad array of objects based on image-level supervision. Despite DETIC's expansive detection capabilities, its application in an egocentric perspective is challenged by temporal inconsistencies, leading to fluctuating predictions across consecutive frames.

To address this, we integrated **Spatio-Temporal Adapters (ST-Adapters)** into the DETIC framework, aiming to incorporate the model with a spatio-temporal awareness that maintains detection consistency over time without sacrificing its open-vocabulary proficiency. The goal is to achieve reliable object detection in a 3D simulation, directly impacting the robot's ability to navigate and interact within complex, ever-changing domestic spaces.

Our methodology encompasses the generation of test sets from the **Habitat Synthetic Scenes Dataset (HSSD)**, where the *in-domain test sets* consist of the same household scenes as those used in the training and validation phases. Conversely, the *out-of-domain test sets* comprise entirely different household scenes, not seen during training or validation, to assess the model's generalization capabilities in unfamiliar environments.

To provide a more clear understanding of the issues with temporal consistency in object detection, we examine an example featuring two sets of four RGB frames each, representing consecutive timesteps and capturing the same scene from a home environment. The scene depicts a bedroom, where we can observe a bed with cushions and a blanket, a potted plant, and a shelf with various items. These frames represent typical data that a robot would encounter as it navigates through a room. Small differences in appearance from frame to frame—due to the robot's movement or changes in camera angle—can lead to errors in the detection models, particularly when the same object appears slightly different at each timestep. This often results in "noisy" predictions, where the model's interpretations of what is in each frame vary from one moment to the next, even though the environment has not significantly changed.

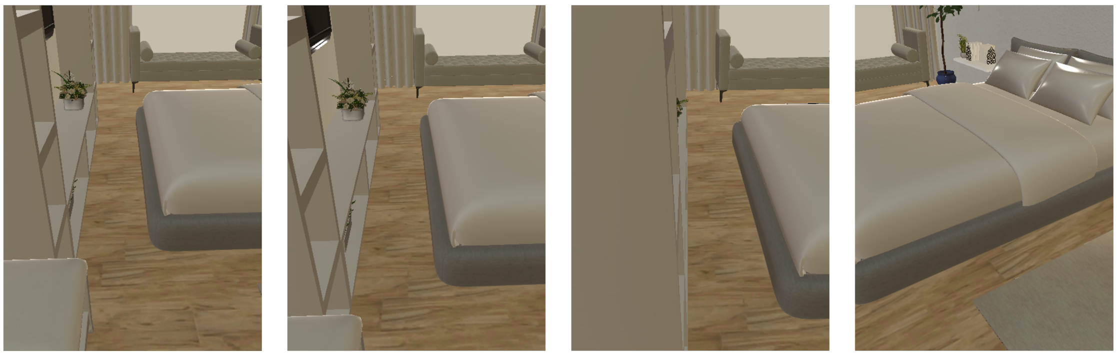


Figure 3.1: First set of consecutive timestep RGB frames of a bedroom scene, arranged from left to right.

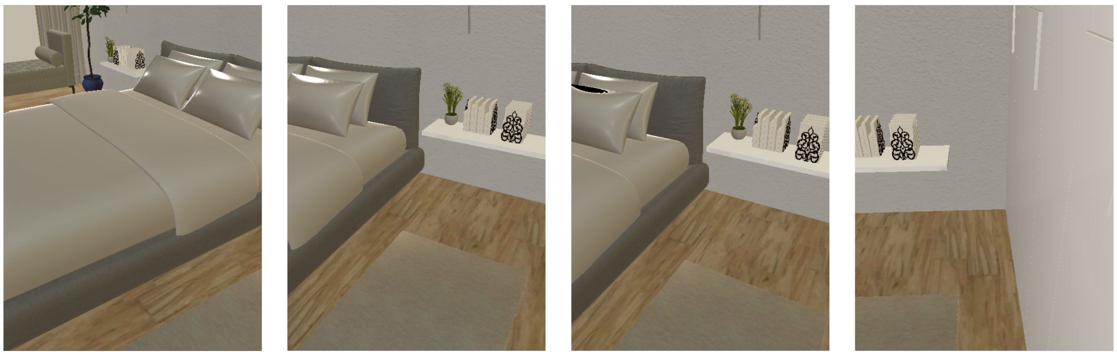


Figure 3.2: Second set of consecutive timestep RGB frames of the same bedroom scene, arranged from left to right.

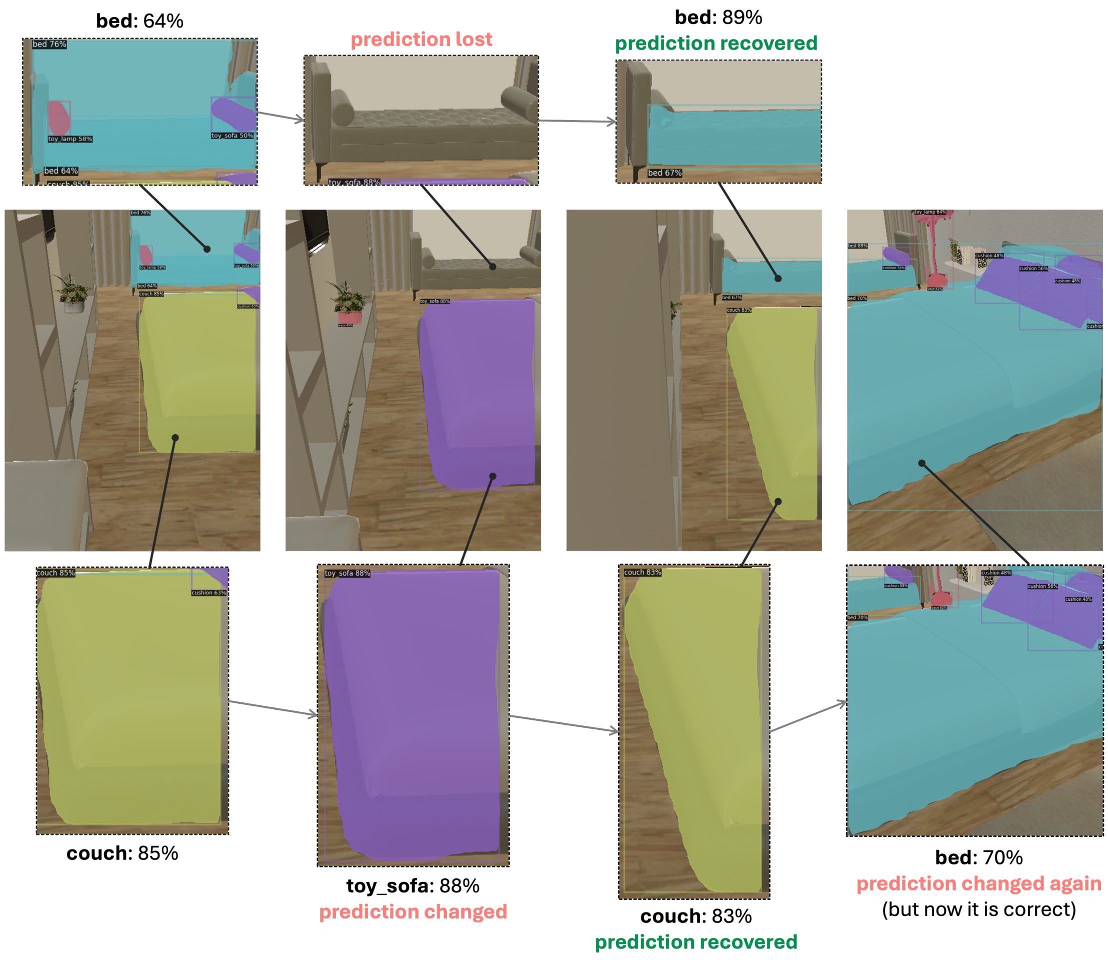


Figure 3.3: Sequence of DETIC predictions over the first set of consecutive frames, demonstrating temporal inconsistencies in object detection.

In examining the predictions of DETIC for the first set of frames, we observe a clear illustration of the challenges related to temporal consistency. The sequence shows how the model's certainty in identifying objects fluctuates. Initially, certain elements in the room, such as a `bed` and `couch`, are identified. However, as we progress to the next frames, the model momentarily loses the predictions (*prediction lost*) or changes them (*prediction changed*), due to the change in viewing angle or occlusion. In the initial frame, the model identifies a `bed` with 64% confidence. Moving to the next frame, this prediction vanishes entirely, indicating a lost prediction of the `bed`. Subsequently, the model regains confidence and identifies the `bed` again, now with a higher certainty of 89%. Similarly, the model initially recognizes a `couch` with an 85% confidence level, but in the following frame, it incorrectly labels it as a `toy_sofa` at 88%. The third frame shows a reversion back to `couch` at 83% confidence, with ongoing mislabeling. In the fourth frame, the model changes its prediction to the correct `bed` class with a 70% confidence score.

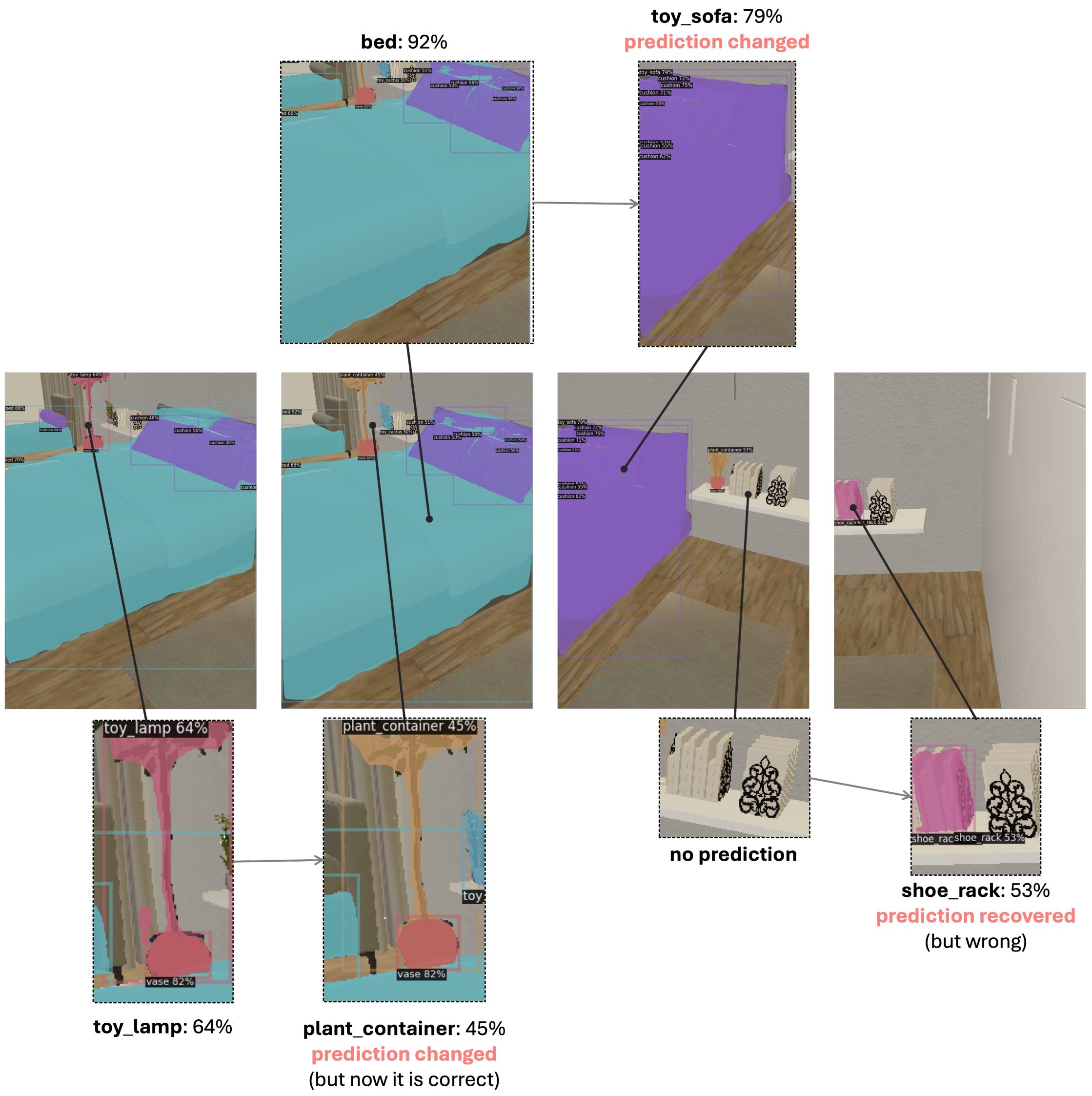


Figure 3.4: Sequence of DETIC predictions over the second set of consecutive frames, demonstrating temporal inconsistencies in object detection.

In the second set of frames analyzed, the challenges of model consistency are further demonstrated. The first frame shows an object incorrectly identified as `toy_lamp` with 64% confidence. In the subsequent frame, the model adjusts its prediction to `plant_container` at 45% confidence, correcting the previous mistake. Meanwhile, the bed is initially detected with high certainty at 92%, but in the third frame, the prediction erroneously shifts to `toy_sofa` with 79% confidence. The third frame results in no prediction for an object that was previously identified, leading to the fourth frame where a completely new (incorrect) prediction emerges as `shoe_racket` with 53% confidence.

3.2 Integrating Spatio-Temporal Adapters in DETIC

To mitigate temporal inconsistencies in object detection, we integrate Spatio-Temporal Adapters (ST-Adapters) into the DETIC framework. The architecture of these adapters has been detailed in chapter 2.6.2.

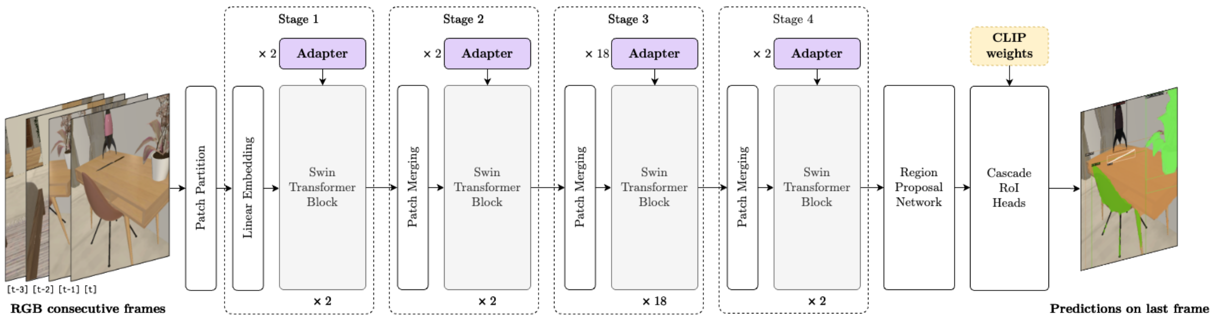


Figure 3.5: Architecture of DETIC with integrated ST-Adapters for improved temporal consistency in object detection.

The design of ST-Adapters is systematic, composed of an upsample process, followed by a three-dimensional convolution (conv3d), and concluded with a downsample step. The role of each is as follows:

- **Upsample**: This step increases the feature maps' resolution, making the spatial dimensions suitable for the next stage of processing.
- **Conv3D**: This is the central element for temporal data integration. It extends beyond 2D convolutions by processing data across consecutive frames, thus capturing the object's temporal behavior and motion. This enables the model to detect patterns over time, enhancing the detection stability across frames.
- **Downsample**: The last step reduces the resolution of feature maps after temporal processing, which simplifies the data and focuses on the most relevant features for object detection.

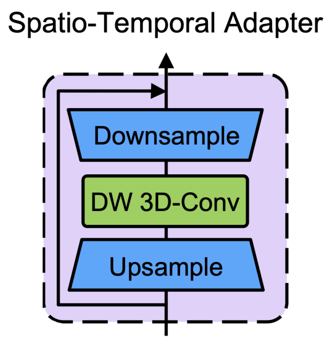


Figure 3.6: Schematic representation of the Spatio-Temporal Adapter (ST-Adapter) architecture, demonstrating the sequential process of upsample, 3D convolution, and downsample operations to enhance temporal feature integration. Image adapted from *ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning* [72].

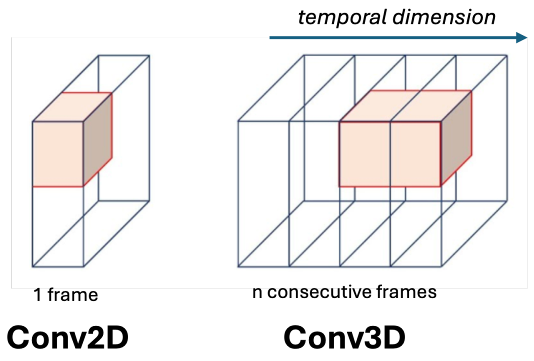


Figure 3.7: Comparison of 2D and 3D convolution operations. The left image illustrates a single 2D convolutional layer processing spatial information in isolation, while the right image shows a 3D convolutional layer that processes both spatial and temporal data, integrating information across consecutive frames as indicated by the arrow direction

The adapter's input is a tensor X with dimensions (BT, L, C), where B represents the batch size, T the number of consecutive frames, L the feature length per frame, and C the number of channels. This configuration indicates that each batch contains a series of elements, each comprised of T related consecutive frames, thereby forming a continuous segment over time. This setup allows the adapter to process a sequence of frames collectively, preserving the temporal continuity among them.

This reconfiguration of the input tensor X from (BT, L, C) to explicitly include temporal dimensions, and subsequently its processing through the adapter, is designed to ensure compatibility with the original model architecture. The Swin Transformer, which forms the backbone of the model, typically receives input with dimensions (B, L, C), where each batch B consists of single frames. Adapting the input in this manner allows the model to incorporate sequences of frames as if they were single frames, thus enabling the analysis of temporal information within what is originally a spatial processing framework.

During the forward pass, the input tensor X is initially transformed into a sequence of batched images, each set of T consecutive frames treated as a separate group within the batch. This step aligns with the data organization during training, facilitating the adapter's interpretation of temporal dynamics across each frame series.

Algorithm 1 Adapter Forward Pass

```
Require: Input tensor $x$ with dimensions $(BT, L, C)$, height $H$, width $W$
Ensure: Output tensor with incorporated spatio-temporal information
  procedure AdapterForward($x$, $H$, $W$)
      $T \leftarrow$ Number of frames per sequence
      $B \leftarrow BT/T$
      $C_a \leftarrow$ Adapter channels
      $x_{\text{id}} \leftarrow x$                              ▷ Preserve the original input tensor
      $x \leftarrow \text{FC1}(x)$                                               ▷ Upsample
      $x \leftarrow x.\text{view}(B, T, H, W, C_a)$
      $x \leftarrow x.\text{permute}(0, 4, 1, 2, 3)$                ▷ Reorganize for 3D convolution
      $x \leftarrow \text{Conv3D}(x)$                                    ▷ Apply 3D convolution
      $x \leftarrow x.\text{permute}(0, 2, 3, 4, 1).\text{view}(BT, L, C_a)$
      $x \leftarrow \text{FC2}(x)$                                             ▷ Downsample
      $x_{\text{output}} \leftarrow x_{\text{id}} + x$                              ▷ Residual connection
      return $x_{\text{output}}$
  end procedure
```

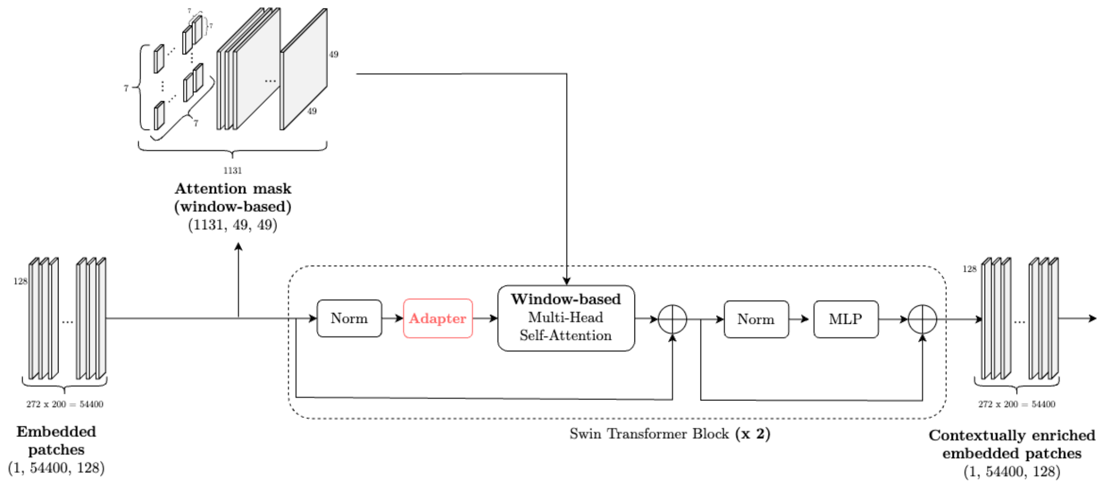


Figure 3.8: The Spatio-Temporal Adapter, highlighted in red, is integrated within the Swin Transformer blocks, following the normalization layer and preceding the multi-head self-attention mechanism. This placement is crucial for the temporal enhancement of feature maps across sequential data.

The integration of the Spatio-Temporal Adapter within the Swin Transformer blocks is aimed at enhancing the model's capability to process sequences of data that contain both spatial and temporal information. Positioned after the normalization layer within each Transformer block, the adapter is set to modify the intermediate representations before they are passed to the window-based multi-head self-attention mechanism.

The adapter enriches the feature maps with temporal context, which is crucial for tasks requiring an understanding of object continuity and motion over time. The temporal aspect becomes particularly important when dealing with video or any sequential data where subsequent frames carry information that influences the current state.

The design choice to include the adapter within every Transformer block ensures that the temporal information is not merely appended at one level but is instead considered in all the feature extraction process. This allows the model to build upon the temporal information iteratively, layer by layer, creating a more nuanced and temporally aware representation at every stage.

By incorporating the adapter, the model processes each frame considering the sequence's history, recognizing that each frame is supposedly connected and influences the next. The insertion of the adapter, therefore, is supposed to provide the Transformer with an enhanced ability to understand the temporal dynamics of the data, leading to more consistent and reliable object detection across frames.

3.3 Open Vocabulary Data Gathering

In the process of refining the Spatio-Temporal Adapters, it was necessary to train the weights of these adapters. As a proof of concept, we utilized the Habitat Synthetic Scenes Dataset (HSSD), referenced in 2.4.6. The data collection was conducted by deploying the open vocabulary manipulation agent, outlined in 2.4.4, within the 3D HomeRobot environment based on HSSD. As the agent navigated this space, we systematically captured screenshots at each timestep, effectively transforming the rich 3D environmental data into a structured 2D dataset.

Each captured frame was annotated with ground truth data available from within the environment. This privileged information included both the segmentation masks and the corresponding class names for each object in view. The collected dataset, therefore, consists of a series of frames with detailed annotations that identify each object and delineate its boundaries, for a comprehensive basis for training the adapter layers.

For the Open-Vocabulary Mobile Manipulation (OVMM) challenge, the agent was tasked with recognizing and handling objects from a list of **150 different classes**. These classes, detailed in Appendix A.1, span a range of everyday items and receptacles found within a home. The frames in the dataset were labeled with these 150 classes, ensuring that the model could learn to identify and interact with the broad spectrum of objects it might encounter in the challenge. This detailed labeling across numerous classes forms the basis for the comprehensive training of the Spatio-Temporal Adapters in DETIC.

During this data gathering approach for the Spatio-Temporal Adapters, we encountered challenges due to **discrepancies** between the ground truth annotations within the 3D environment and the predefined classes set for detection. The process of extracting accurate ground truth data had involved reverse engineering the environment's code to obtain annotations. However, these annotations were often inconsistent with the defined classes, leading to ambiguity.

This behavior is to be expected, as the HSSD environment was not designed with the OVMM challenge in mind. Given that DETIC needs to operate within the confines of its 150-class vocabulary, the model is constrained to only recognizing those classes. The environment's object classes did not align perfectly with the OVMM classes, resulting in such discrepancies between the ground truth annotations and the model's predictions. This is important for evaluating the model's performance, as the ground truth labels should supposedly matched with the model's predictions to assess its effectiveness.

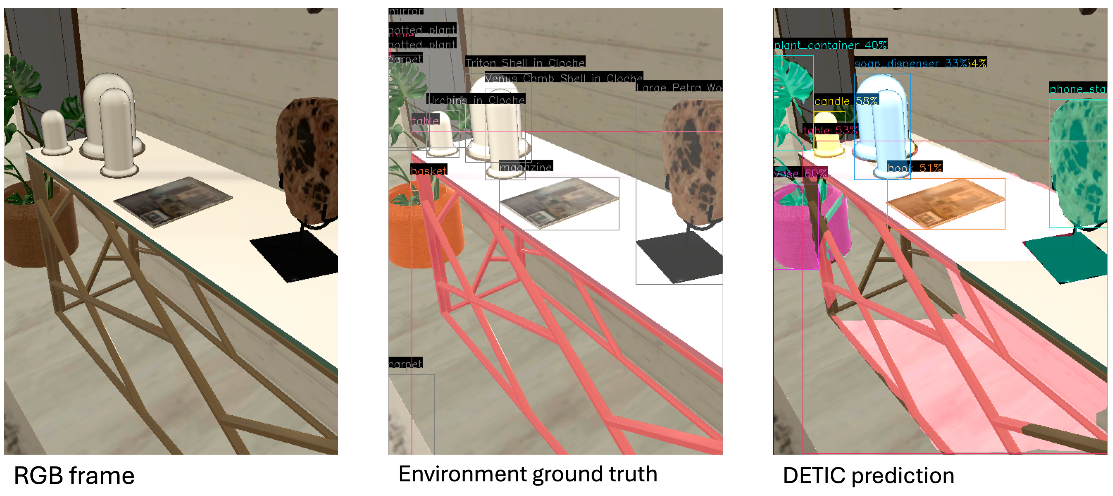


Figure 3.9: Comparison between ground truth annotations and DETIC predictions, highlighting some inconsistencies between ground truth annotations and the classes defined in the OVMM Challenge.

In the figure, the comparison between DETIC predictions and ground truth annotations illustrates key points about handling object classifications within the 150 predefined classes. When DETIC correctly identifies `table`, it aligns with the classes. The misclassification of `basket` as `vase` poses a question of semantic closeness and whether such predictions should be deemed partially correct. DETIC's prediction of `book` for `magazine`—not in the 150 classes—suggests a reasonable semantic match. Similarly, `potted_plant` being predicted as `plant_container` should be considered a valid prediction due to the semantic overlap.

To mention other examples, when the ground truth is labeled as `"table_lamp"` (not included in our vocabulary) and the model predicted `"lamp"` (included in our vocabulary), such a prediction should be deemed partially correct due to the overlap in semantic meaning despite the discrepancy in terminology. Also, despite phonetic differences between words, the labels may share the same semantic meaning. An example involves a model predicting `"towel"` when the ground truth is labeled as `"rag"`.

In the context of Open-Vocabulary Segmentation (OVS), assessing the match between predicted and true categories transcends simple binary evaluations of right or wrong. To navigate this complexity, we can employ two principal methodologies: similarity assessments through BERT embeddings and Path Similarity via WordNet. For predictions analyzed with **BERT**, we apply cosine similarity in the embedding space, considering a prediction as a true positive (**TP**) if the similarity exceeds a certain threshold. This approach appreciates the spectrum of accuracy in predictions, recognizing the value of close matches based on contextual resemblance.

On the other hand, **WordNet**, with its extensive lexical database of 82,115 noun synsets, provides a structured framework for semantic analysis. The **Path Similarity** technique within WordNet offers four key advantages for open-vocabulary tasks:

- It yields moderate similarity scores, avoiding extreme valuations.
- It is capable of distinguishing polysemy, recognizing multiple meanings of words.
- It remains unbiased by training data, offering a neutral evaluation ground.
- It closely aligns with human cognitive processes, reflecting natural language understanding.

The preference for the **WordNet's Path Similarity** method is supported by by the paper *Rethinking Evaluation Metrics of Open-Vocabulary Segmentaion* [74]. WordNet is a large lexical database of English, where nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by semantic and lexical relationships. The Path Similarity method within WordNet quantifies how similar two word senses are, based on the shortest path that connects their synsets in the semantic space. This distance is inversely proportional to semantic similarity: the shorter the path, the more similar the words. Through this method, we aimed to establish a coherent set of Open Vocabulary metrics that accommodate the semantic variance in object labeling.

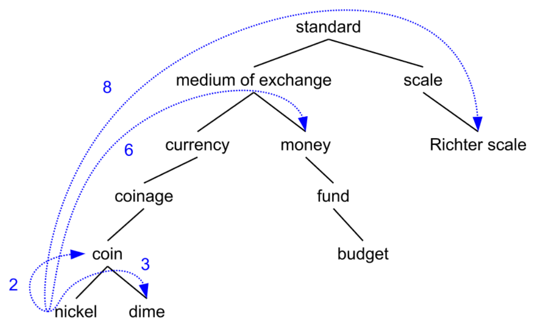


Figure 3.10: An illustration of the WordNet lexical database hierarchy showcasing the semantic relationship between words. Notably, the diagram also exemplifies an erroneous link (indicated by a dotted line), demonstrating how 'Richter scale', which belongs to a different semantic field, can sometimes be mistakenly associated with 'standard', illustrating the complexity and challenges of automated semantic classification.

The provided algorithm is a systematic approach designed for enriching object detection datasets with open vocabulary annotations. The process begins with the input of a list of scenes S, a set of 150 challenge classes C identified for the Open Vocabulary Mobile Manipulation (OVMM) challenge, and a predefined similarity threshold θ. The intended output is a collection of annotated RGB frames. These annotations either directly correspond to one of the OVMM challenge classes or to a closely related class determined by semantic similarity. Eventually, such annotations will serve as a dataset for training the Adapters in the DETIC model.

Algorithm 2 Open Vocabulary Data Gathering

```
 1: **Input:** Scene list $S$, OVMM 150 challenge classes $C$, Similarity Threshold $\theta$
 2: **Output:** Annotated frames with classes from $C$ or closely related classes based
    on similarity
 3: for each scene $S_i$ do
 4:     for each episode $E_j$ in $S_i$ do
 5:         while agent navigates $E_j$ do
 6:             save the rgb frame $F_k$
 7:             retrieve masks and ground truths $\{(M_l, G_l)\}$
 8:             for each $(M_l, G_l)$ do
 9:                 if $G_l$ in $C$ then                ▷ Immediate match in OVMM classes
10:                     save $(M_l, G_l)$
11:                 else              ▷ Find closest class based on WordNet similarity
12:                     $sim_{max} \leftarrow 0$
13:                     $C_{best} \leftarrow$ null
14:                     for each class $C_m$ in $C$ do
15:                         $sim \leftarrow$ WordNetPathSimilarity($G_l$, $C_m$)
16:                         if $sim > sim_{max}$ then
17:                             $sim_{max} \leftarrow sim$
18:                             $C_{best} \leftarrow C_m$
19:                         end if
20:                     end for
21:                     if $sim_{max} > \theta$ then
22:                         save $(M_l, C_{best})$
23:                     end if
24:                 end if
25:             end for
26:         end while
27:     end for
28: end for
```

For each scene within the scene list, the algorithm iterates through each generated episode. Within these episodes, as an agent navigates the environment, it performs multiple tasks at each timestep: it saves the current RGB frame and retrieves the associated object masks along with their ground truth labels. For each mask and its corresponding label, the algorithm checks if the label is among the predefined OVMM challenge classes. If so, the pair (mask and label) is saved directly.

In cases where the ground truth label does not match any of the predefined classes, the algorithm employs a semantic similarity measure, specifically the WordNet path similarity, to compare the label against each of the 150 challenge classes. It selects the class with the highest similarity score. If this score exceeds the similarity threshold θ, indicating a strong semantic relation, the algorithm saves the mask with the most closely related challenge class as its new label.

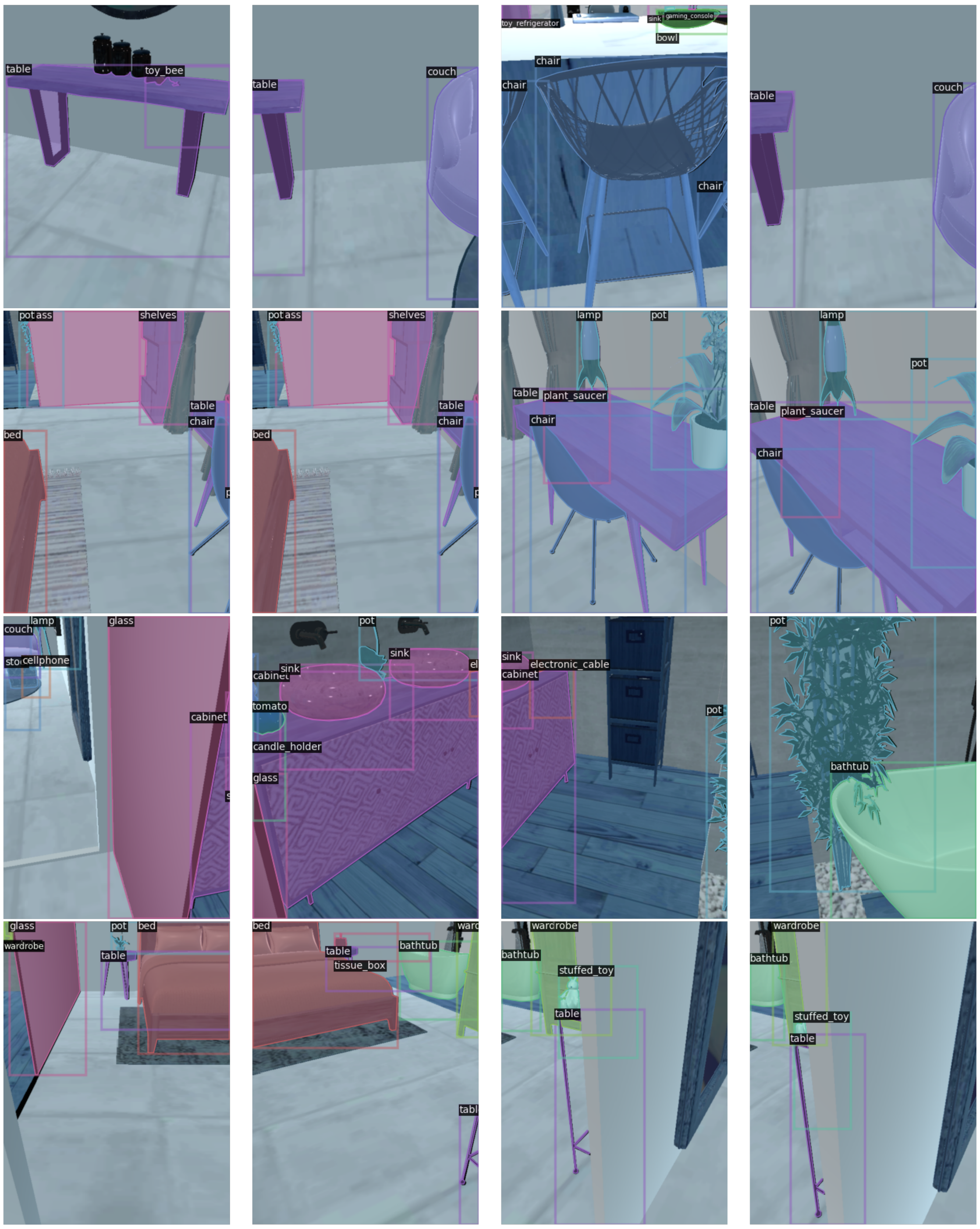


Figure 3.11: Examples of frame successions annotated with objects and receptacles ground truths, showcasing the complexity of the scenarios encountered in the 3D environment.

3.4 Dataset Splits

For the data collection methodology of this research, we employed an egocentric perspective to capture frames as the agent navigated through various environments using the HomeRobot framework. This setup involved twelve distinct scenes from the Habitat Synthetic Scenes Dataset (HSSD), each representing a unique household environment. Within these environments, a total of 1,200 episodes were conducted—100 episodes per scene—where each episode featured a different, randomly determined starting point and a distinct arrangement of objects within the scene. Throughout each episode, the agent traversed the environment for approximately **120 steps**, yielding around **120 frames per episode** to capture the diversity and complexity of household navigation and object interaction.

The dataset was divided into several splits to facilitate comprehensive training, validation, and testing of the models:

- 9 *in-domain scenes* (accounting for 900 episodes), simulating environments that the models were expected to become familiar with during their training. From this subset:
  - 70% (approximately 630 episodes) for the *training set*,
  - 10% (about 90 episodes) for the *validation set*,
  - 20% (circa 180 episodes) for the *in-domain test set*.
- 3 *out-of-domain scenes* (comprising 300 episodes), to challenge the models with previously unseen environments and assess their generalization capabilities, featuring households that were not encountered by the models during training.

In terms of dataset composition:

- The *training split* contained an estimated 61,7k images and 300k annotations across 630 episodes.
- The *validation split* contained an estimated 8,7k images and 41k annotations across 90 episodes.
- The *in-domain testing* contained an estimated 18k images and 88,5k annotations across 180 episodes.
- Lastly, the *out-of-domain test split* contained an estimated 31,4k images and 271k annotations across 300 episodes set in unfamiliar environments.

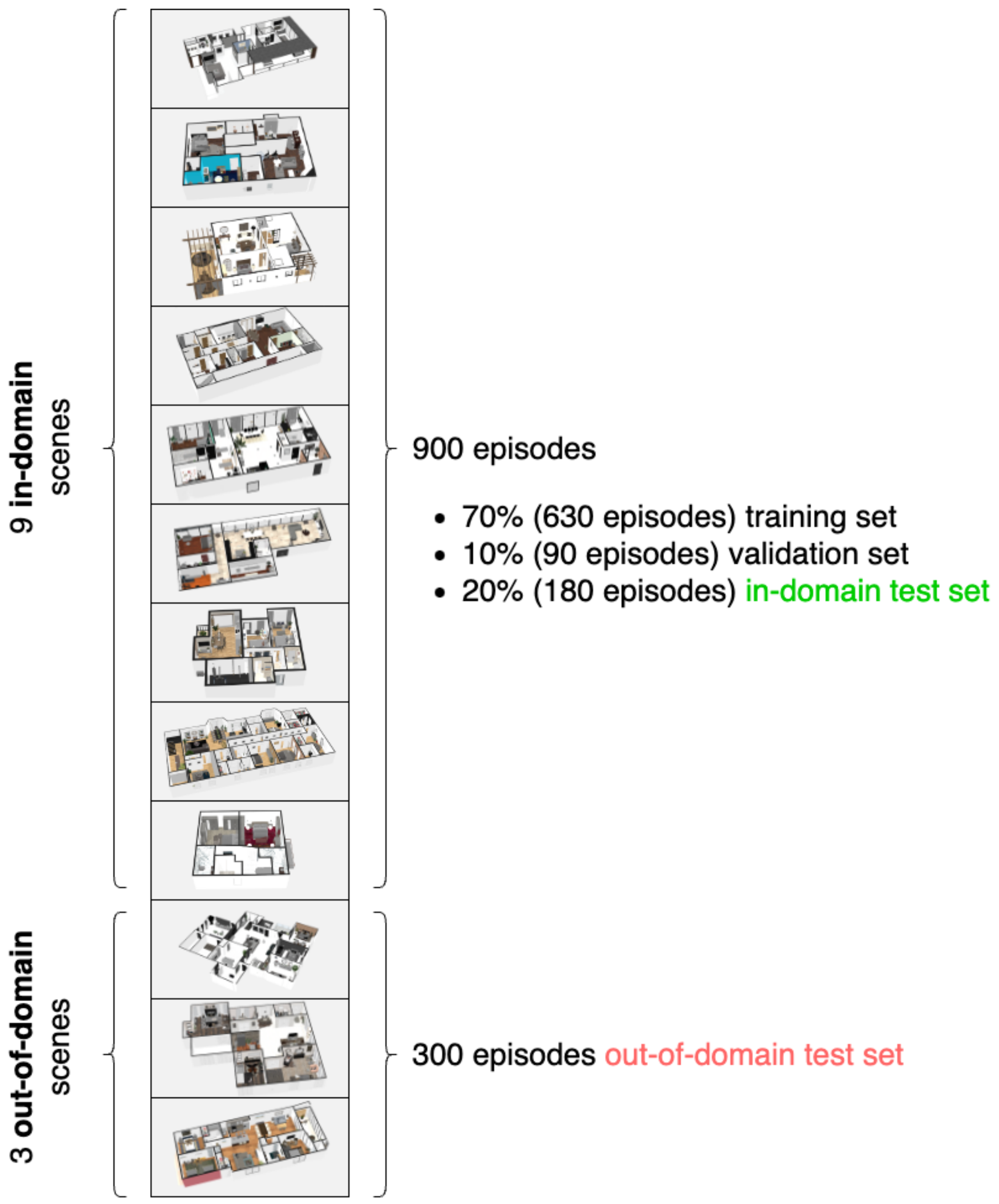

9 in-domain scenes
900 episodes
• 70% (630 episodes) training set
• 10% (90 episodes) validation set
• 20% (180 episodes) in-domain test set
3 out-of-domain scenes
300 episodes out-of-domain test set

Figure 3.12: Overview of dataset splits for model evaluation. This figure depicts the division of the scenes into various splits, providing a structured approach to training, validation, and testing.

Chapter 4

Implementation Details

We now delve into the technical aspects of the implementation, focusing on the key components and methodologies employed to enhance the model's performance in temporal consistency. This chapter provides a detailed overview of the techniques used to prepare the model for training, the evaluation metrics employed, and the modifications made to adapt the model to handle multiple frames. This methodology is implemented within Detectron2 [75], a popular deep learning framework developed by Facebook AI Research. The framework offers a flexible and modular structure, enabling the integration of our custom components and functionalities to enhance the model's capabilities.

4.1 Detectron2 Custom Dataset

Integrating the handling of multiple consecutive frames into Detectron2 required extending its core components to support temporal data, for object detection in video sequences. The original Detectron2 framework is designed around processing single images; thus, our modifications introduce a new approach to process sequences of images, considering them as unified entities during both training and inference. This is achieved through the introduction of specialized classes derived from existing Detectron2 components, which have been adapted to manage sequences of frames.

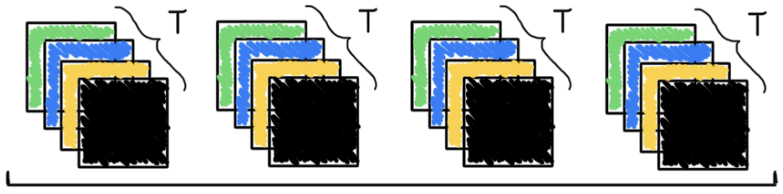


Figure 4.1: A sketch of a batch comprising 4 different and unrelated series of frame sequences. Each column represents a batch element containing a sequence of frames (T), where T denotes the number of consecutive frames.

The `ConsecutiveFramePredictor` class is an extension of Detectron2's predictor functionality, designed to handle multiple input images as a series of consecutive frames. Unlike the original predictor, which processes single images, this class accepts a list of images and processes them together, maintaining their temporal relationship. This modification enables the model to consider temporal information when making predictions, crucial for tasks such as object tracking and activity recognition in video streams.

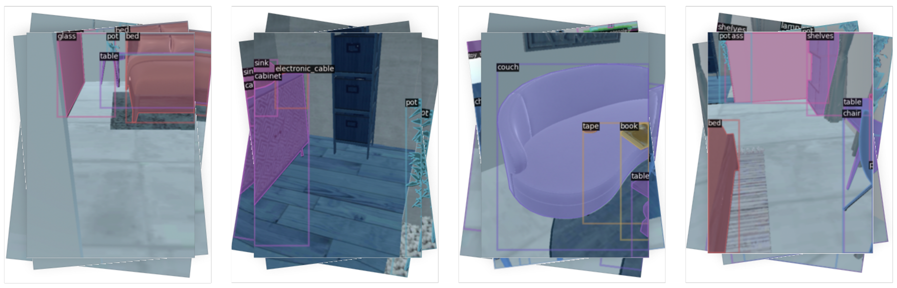


Figure 4.2: A batch comprising 4 different and unrelated series of frame sequences. These sequences are independent of each other and are processed collectively as one batch in the model. The diversity within the batch reflects different temporal contexts, which are crucial for the model to learn from varying scenarios.

Similarly, the `ConsecutiveFrameDataset` class modifies the way datasets are handled. Derived from Detectron2's `DatasetFromList`, this class is tailored to load and return groups of consecutive frames rather than individual images. By doing so, it aligns the dataset's structure with the temporal nature of the tasks at hand, ensuring that the model receives appropriately formatted input. This is particularly important for training models on video data, where the temporal context of an object's appearance and movement can provide additional insights beyond static images.

The `ConsecutiveFrameDatasetMapper` serves as a bridge between the dataset and the model, transforming the raw data into a format suitable for model consumption. As an adaptation of the `DatasetMapper` in Detectron2, it applies necessary transformations to each frame in a sequence, such as resizing, normalization, and augmentation. Furthermore, it ensures that the frames are correctly batched together, preserving their temporal order. This class plays a crucial role in preparing the data for the model, ensuring that the temporal information contained in sequences of frames is effectively captured and utilized during both training and inference.

4.2 Weights Initialization

During the initial phase of preparing the model, particular attention is given to initializing the weights of the adapter layers. This step is crucial because it sets the starting point for how these layers learn and adapt during training. We employ two main methods for this initialization:

- **For Convolutional 3D (`nn.Conv3d`) layers**, we use the **Kaiming Normal** method. This method is chosen because it suits layers that use the ReLU activation function. The idea here is to initialize the weights in a way that helps prevent the gradients from becoming too small or too large as they backpropagate through the network.
- **For Linear (`nn.Linear`) layers**, the **Xavier Uniform** initialization is applied. This method adjusts the scale of the weights based on the number of input and output neurons, aiming for a uniform distribution of weights. This balance is important for maintaining a stable gradient flow across layers, especially at the start of training, making it easier for the model to learn.

After initialization, the model's parameters are updated with the pre-trained weights. This process allows the model to retain its previously acquired knowledge, serving as a foundation for learning new information.

A crucial aspect of this preparation phase is the **selective freezing of the model's parameters**, with the exception of those within the adapter layers. Gradients are disabled for all parameters except those in the adapter layers. This approach preserves the model's pre-existing knowledge while focusing training on improving the adapters' capability to understand and manage spatio-temporal data.

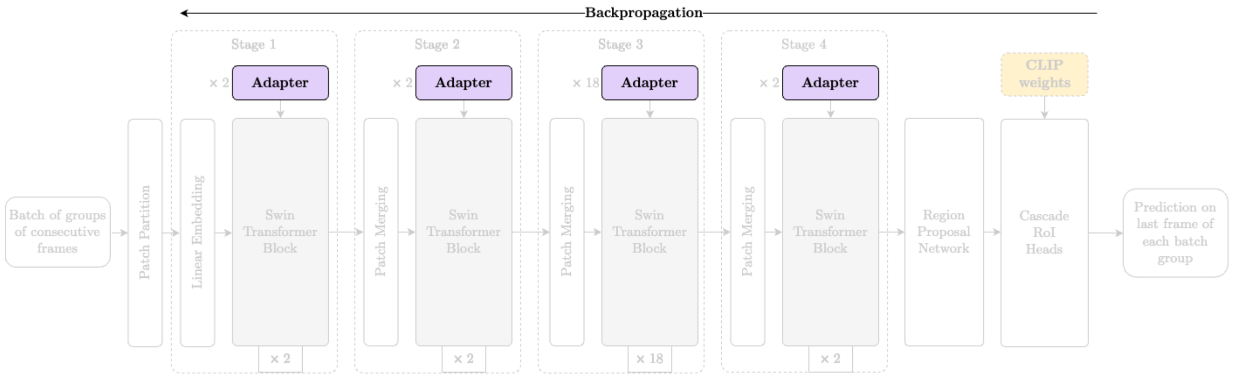


Figure 4.3: Selective freezing of the model's parameters, with the exception of those within the adapter layers. Gradients are disabled for all parameters except those in the adapter layers.

The last step sets the model to **training mode**, activating functions such as dropout and batch normalization, which are important during training but not used during model evaluation. Identifying which parameters can change during this phase highlights the beginning of a targeted training, aiming to improve the model's handling of spatio-temporal data while trying to mantain its original object detection performance.

Algorithm 3 Prepare Model for Training

```
 1: procedure PrepareModel(model, config)
 2:     // Initialize weights of the adapter layers
 3:     for each name, module in model.named_modules() do
 4:         if "adapter" in name then
 5:             if module is nn.Conv3d then
 6:                 Initialize module.weight with Kaiming Normal
 7:             else if module is nn.Linear then
 8:                 Initialize module.weight with Xavier Uniform
 9:                 if module.bias is not None then
10:                     Initialize module.bias with Zeros
11:                 end if
12:             end if
13:         end if
14:     end for
          model.weights ← weights from cfg.MODEL.WEIGHTS
15:     // Disable gradients for all parameters
16:     for each param in model.parameters() do
17:         Set param.requires_grad to False
18:     end for
19:     // Enable gradients only for parameters in adapter layers
20:     for each name, param in model.named_parameters() do
21:         if "adapter" in name then
22:             Set param.requires_grad to True
23:         end if
24:     end for
25:     // Set the model to training mode
26:     model.train()
          return model
27: end procedure
```

4.3 Evaluation Metrics Overview

In the field of object detection and segmentation, the metrics employed to evaluate model performance can significantly vary depending on the vocabulary scope—namely, whether the evaluation is conducted in a **closed vocabulary** or an **open vocabulary** context:

- **Closed vocabulary** evaluations focus on a predefined set of classes known to the model during training, offering a controlled environment to measure precision and recall.
- **Open vocabulary** evaluations represent a more challenging and dynamic scenario where the model encounters objects and classes not seen during training, necessitating metrics that can effectively capture the model's adaptability and generalization capabilities.

This work primarily employs **open vocabulary metrics** to assess the performance of object detection and segmentation tasks. This approach aligns with the discussed challenges of robotic navigation and interaction within home environments, where a robot may encounter a objects beyond those it was explicitly trained on. Open vocabulary metrics, including modifications to the classical mean Intersection over Union (mIoU) and the introduction of similarity metrics, allow us to evaluate the model's ability to generalize its detections to previously unseen objects, providing a comprehensive view of its performance in realistic and dynamic settings.

4.3.1 Mean Intersection over Union (mIoU)

In evaluating segmentation tasks with a closed vocabulary, the classical Intersection over Union (IoU) is a fundamental metric. This metric measures the precision of predictions compared to the ground truth. Predicted results are categorized into three groups:

- **True Positives (TP)**: Instances where the model correctly predicts the presence of a class.
- **False Positives (FP)**: Instances where the model incorrectly predicts the presence of a class.
- **False Negatives (FN)**: Instances where the model incorrectly predicts the absence of a class.

The IoU is calculated as $\text{IoU} = \frac{|TP|}{|TP|+|FP|+|FN|}$, where |TP|, |FP|, and |FN| indicate the numbers of pixel-level true positives, false positives, and false negatives, respectively. Therefore, the mIoU is defined as:

$$\text{mIoU} = \frac{1}{k+1}\sum_{i=0}^{k}\text{IoU}_i, \text{ where } \text{IoU}_i = \frac{|TP|}{|TP|+|FP|+|FN|}$$

where IoU_i represents the IoU for class i, and $k+1$ is the total number of classes in the dataset being evaluated. This measure provides an overview of the model's segmentation accuracy across different classes and object sizes.

As discussed in 3.3, the traditional metrics overlook the significance of the similarity between the predicted and actual categories. For instance, in a scenario where the ground truth is `'table'`, the conventional methodology would assign a zero true positive (TP) score to predictions labeled as `'cabinet'` or `'chest'`. Such an approach contradicts the intuitive human understanding that these predictions should not be outright dismissed. This discrepancy highlights the necessity for a revised evaluation framework that more accurately gauges a model's performance in open-world settings, serving as the foundation for our exploration into more appropriate methods for assessing open-vocabulary segmentation models.

In the context of evaluating Open-Vocabulary Segmentation (OVS), the concept of mean Intersection over Union (mIoU) is adapted to accommodate the open vocabulary paradigm, incorporating nuanced definitions of True Positives (TP), False Positives (FP), and False Negatives (FN) based on semantic similarities. This adaptation, referred to as Open mIoU, integrates the flexibility of open vocabulary by leveraging similarity scores between predicted and ground truth categories.

The Open mIoU calculation maintains the core principle of mIoU but refines the criteria for TP, FP, and FN to reflect the open vocabulary setting. For instance, in a scenario where the ground truth label c_i differs from the predicted label c_j, the assignment of TP, FP, and FN is governed by the similarity score $S_{c_i c_j}$ between the two categories. Such similarity score is based on the Path Similarity via WordNet, as discussed in in 3.3. The refined definitions are as follows:

- **Soft True Positive (TP)**$_{c_i}$: The similarity score $S_{c_i c_j}$ acts as a soft measure for TP, quantifying the degree of correctness in the prediction relative to the ground truth category.
- **False Positive (FP)**$_{c_j}$: Defined as $1 - S_{c_i c_j}$, this metric captures the portion of the prediction that does not align with the ground truth, considering the semantic gap between c_i and c_j.
- **False Negative (FN)**$_{c_i}$: Similarly, $1 - S_{c_i c_j}$ is used to quantify the missed detection of the ground truth category by the model, taking into account the semantic discrepancy.

This approach to Open mIoU enables a more nuanced evaluation of model performance in OVS tasks, accounting for the inherent variability and richness of open vocabularies. It recognizes that not all mismatches between predictions and ground truth are equally erroneous, allowing for partial correctness based on semantic proximity. This method aligns with the goal of assessing model effectiveness in scenarios where the exact category match is less critical than the conceptual and semantic alignment between prediction and ground truth.

Algorithm 4 Open mIoU Calculation for Open-Vocabulary Segmentation

```
1: **Input:** Predicted Regions $P$, Ground Truth Regions $GT$, Similarity Function $Sim$
2: **Output:** mIoU
3: Initialize $Matches \leftarrow [\,]$
4: Initialize $AssociatedGT \leftarrow \emptyset$        ▷ Track associated ground truths
5: for each $p \in P$ do
6:     $bestIoU \leftarrow 0$
7:     $bestMatch \leftarrow None$
8:     for each $gt \in GT$ do
9:         if $gt$ not in $AssociatedGT$ then
10:            $simScore \leftarrow Sim(p\_class, bestMatch\_class)$
11:            $IoU \leftarrow$ CalculateIoU($p$, $gt$, $simScore$)
12:            if $IoU > bestIoU$ then
13:                $bestIoU \leftarrow IoU$
14:                $bestMatch \leftarrow gt$
15:            end if
16:        end if
17:    end for
18:    if $bestMatch \neq None$ then
19:        $AssociatedGT \leftarrow AssociatedGT \cup \{bestMatch\}$
20:        $Matches.append((p, bestMatch, bestIoU))$
21:    end if
22: end for
23: $IoU \leftarrow$ Mean of all calculated $IoU$ values
24: return $mIoU$
```

The algorithm provided outlines the process for calculating the Open mean Intersection over Union (mIoU) during the evaluation of a segmentation model in an open-vocabulary context. The procedure is an enhancement of the traditional mIoU calculation that includes a similarity function to account for the semantic relationship between predictions and ground truths. For each predicted region, the algorithm searches for the best matching ground truth region based on the highest IoU score, ensuring that each ground truth region is uniquely associated with a prediction. The inclusion of the similarity function, *Sim*, allows the algorithm to compute a similarity score alongside the IoU, reflecting the semantic relatedness of the predicted and ground truth classes. This approach rewards predictions that are semantically similar to the ground truth, thus providing a more nuanced assessment of the model's open vocabulary performance. The calculated IoUs for the best matches are then averaged to obtain the mIoU, which is returned as the output of the algorithm. By integrating semantic similarity into the evaluation metric, the algorithm aligns more closely with human judgment, acknowledging the complexity of real-world categorization and the open-vocabulary nature of the segmentation task.

4.3.2 Success Rate over Navigation Episodes

In our setup, the OVMM task is defined by instructions such as: `"Move (object) from the (start_receptacle) to the (goal_receptacle)"`. The specified `object` is typically a small, manipulable household item, while the `start_receptacle` and `goal_receptacle` are larger pieces of furniture capable of holding these items. The robot begins in a single-floor home environment, such as an apartment, with the challenge of locating and moving an `object` from its known starting location on a `start_receptacle` to any designated `goal_receptacle`. Success in this context is not binary; we award partial credit for completing stages of the task: locating the starting receptacle with the `object`, securing the `object`, finding the `goal_receptacle`, and finally placing the `object` appropriately.

Simulation success is detailed across four stages:

- **`FindObj`**: The agent is successful if it reaches a position within 0.1 meters of the target `object` on the `start_receptacle`, ensuring the `object` occupies at least 0.1% of the camera frame pixels.
- **`Pick`**: After successful `FindObj`, the agent must activate its gripper when the `object` instance is in view and within 0.8 meters, upon which the `object` is 'snapped' to the gripper in the simulation.
- **`FindRec`**: After successful `Pick`, the agent must approach within 0.1 meters of a viewpoint where the `goal_receptacle` is visible, similarly with the `object` occupying at least 0.1% of the camera frame pixels.
- **`Place`**: After successful `FindRec`, the agent needs to place the `object` at the `goal_receptacle`, with the `object`'s stability defined by maintaining contact and velocities below specified thresholds for 50 contiguous steps, without any collision events during placement.

An episode is declared successful if the robot completes all four stages within 1250 steps. Evaluation metrics include the number of steps taken (`num_steps`) and success in each phase (`find_object_phase_success`, `pick_object_phase_success`, `find_recep_phase_success`). Overall success (`overall_success`) signifies the robot's competence in executing the complete task, while partial success is computed as the average of success across all individual phases and overall task completion:

$$\begin{aligned}\texttt{partial\_success} = \frac{1}{4}(&\texttt{find\_object\_phase\_success} + \\ &\texttt{pick\_object\_phase\_success} + \\ &\texttt{find\_recep\_phase\_success} + \\ &\texttt{overall\_success})\end{aligned}$$

These metrics offer a comprehensive measure of the robot's proficiency in navigating and interacting within its environment.

Chapter 5

Results

5.1 Global mIOU Performance Analysis

In this section, we conducted a global analysis by focusing on the Global mean Intersection over Union (mIOU), providing a macroscopic view of the models' performance across all object categories and scenarios without delving into specific size bins or object types. We conducted a comparative analysis between the baseline off-the-shelf DETIC model and versions of DETIC enhanced with trained adapters, focusing specifically on variations in the temporal dimension (T) of the adapters. The adapters were examined with $T = 1$, $T = 4$, and $T = 5$.

The $T = 1$ setup restricts the model to spatial reasoning, lacking temporal analysis, and it is used to understand the impact of incorporating historical data. All the model configuration selected for comparison utilized a learning rate of 10^{-5} and a batch size of 4, with the best models chosen at around 90,000 iterations, based on validation performance.

Comparative results highlighted significant improvements in Global mIOU when temporal reasoning was introduced via trained adapters. The following table summarizes these findings:

Model Configuration	**Global mIOU**
DETIC	0.140
DETIC with 2DConv (T = 1)	0.330
DETIC with 3DConv (T = 4)	**0.391**
DETIC with 3DConv (T = 5)	0.372

Table 5.1: Comparison of Global mIOU on the out-of-domain dataset for different DETIC model configurations, illustrating the model's generalization capabilities.

These findings underscore the value of integrating temporal information for improving object detection performance in dynamic environments. The optimal global performance was observed with $T = 4$, suggesting a balanced incorporation of temporal dynamics that significantly exceeds the spatial-only analysis and slightly outperforms the more extended sequence analysis of $T = 5$. This evidence guided our selection of the DETIC model with conv3d adapters trained with $T = 4$ as the preferred model, demonstrating the advantages of temporal reasoning in enhancing the model's comprehension of complex scenes.

The preference for **4 adapter frames** over the other settings also stems from the temporal characteristics of the dataset. Given the temporal distance between frames, a setup of more frames might introduce too much variability, making it challenging for the model to discern consistent patterns. Reducing the adapter frames to 4 enables the model to focus on immediate temporal relationships, and balance computational complexity given by the temporal context.

5.2 Object Size mIOU Performance Analysis

Understanding the model's performance across a variety of object sizes is critical for assessing its robustness and applicability. We included configurations with different temporal dimensions, specifically $T = 1$, $T = 4$, and $T = 5$, as in the previous global analysis. This examination enabled us to assess spatio-temporal **adaptability** of the model across objects of **varying sizes**. Objects of different sizes may present unique challenges in detection and recognition, particularly in dynamic environments where perspective and scale can vary significantly.

To evaluate the performance across varying object sizes, we divided the detection scenarios into **five bins**, based on the objects' pixel dimensions within a `640x480` pixel frame (`307200` pixels). The bins are defined as follows, based on the object's size in pixels:

- **Very Small (0-20%)**: Objects occupying `0` to `61440` pixels in the frame.
- **Small (21%-40%)**: Objects occupying `61441` to `122880` pixels in the frame.
- **Medium (41%-60%)**: Objects occupying `122881` to `184320` pixels in the frame.
- **Big (61%-80%)**: Objects occupying `184321` to `245760` pixels in the frame.
- **Very Big (81%-100%)**: Objects occupying `245761` to `307200` pixels in the frame.

Model Configuration	**Very Small**	**Small**	**Medium**	**Big**	**Very Big**	**Global mIoU**
DETIC	0.143	0.141	0.133	0.116	0.100	0.140
DETIC 2DConv (T = 1)	0.337	0.298	0.268	0.154	0.139	0.331
DETIC 2DConv (T = 4)	0.395	0.324	0.322	0.214	0.176	**0.391**
DETIC 2DConv (T = 5)	0.377	0.320	0.361	0.244	0.200	0.372

Table 5.2: Comparison of Mean IoU per size bin and Global Mean IoU for different DETIC model configurations, showcasing performance across varying object sizes in the out-of-domain dataset.

The results presented in the table show more clearly the performance differences across DETIC model configurations when analyzed through the lens of object size. Notably, the DETIC 2DConv ($T = 4$) configuration outperforms DETIC 2DConv ($T = 5$) in the `"Very Small"` size bin, achieving a Mean IoU of 0.395 compared to 0.377. The improved detection of very small objects using ($T = 4$) can be explained by its effective use of time-related data without focusing too much on repetitive patterns, which are not as common in very small objects. This balance helps the model recognize small details better without getting confused by less consistent information over time.

In contrast, for sizes beyond `"Small"`, particularly in the `"Medium"`, `"Big"`, and `"Very Big"` categories, we observe an inverse relationship where $T = 5$ starts to outperform $T = 4$. This shift is particularly noticeable in the `"Very Big"` size bin, where $T = 5$ achieves a Mean IoU of 0.200 compared to $T = 4$ value of 0.176. Importantly, as the agent gets closer to an object, making it appear larger (or `"Very Big"`), the accumulated spatial-temporal information becomes increasingly significant. The model sees the same object more frequently as it gets closer to it, which enhances its ability to recognize and classify the object correctly due to the accumulation of detailed spatial-temporal features. This mechanism explains why larger objects benefit more from a longer temporal context (T), as the extended analysis captures more of these crucial accumulated interactions.

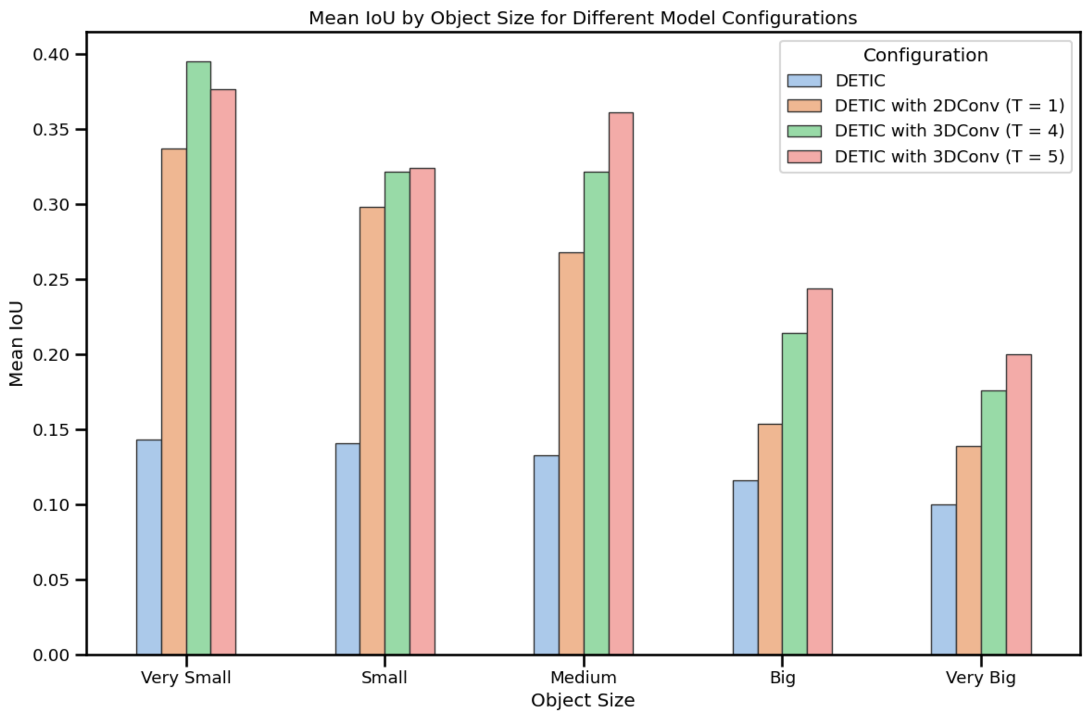


Figure 5.1: Mean IoU per size bin for different DETIC model configurations.

By arranging the results in a histogram (Figure 5.1), we can see a pattern of linear growth in Mean IoU with increasing T values, particularly for larger object sizes. The linear growth in Mean IoU with increasing T values for larger object sizes highlights the incremental value of temporal information in enhancing model performance for objects that dominate the scene. This effect likely stems from the model's ability to aggregate spatial information over time, providing a richer context for identifying and classifying larger objects.

5.3 Qualitative Analysis

In this section, we delve deeper into the practical implications of the findings presented earlier, particularly focusing on a comparative study of the off-the-shelf DETIC model, and the DETIC models enhanced with Conv2D $T = 1$ and Conv3D $T = 4$. As highlighted in Section 3.1, we introduced an example scenario involving a bedroom scene to underscore the challenges associated with achieving temporal consistency in object detection. This scene features everyday objects such as a bed with cushions, a potted plant, and a shelf. In this section, we focus our qualitative analysis on the same bedroom scene.

This qualitative analysis aims to illustrate the differences in model performance, especially in handling the nuances of temporal consistency. The erratic nature of "noisy" predictions, driven by minor variances in object appearance due to the robot's movement or shifts in camera angle, presents a significant challenge. These variances, though seemingly trivial, can drastically affect the detection models' reliability, with the same object being perceived differently across successive frames, despite no real change in the scene.

By comparing the off-the-shelf DETIC model with its adapter (T = 1) and adapter (T = 4) counterparts, we aim to visually demonstrate how the incorporation of temporal information influences the models' ability to maintain consistent object identification over time.

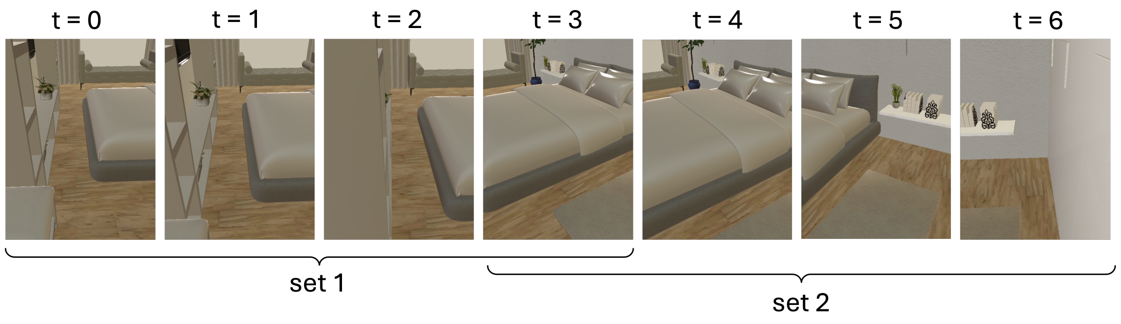


Figure 5.2: The representative bedroom scene for qualitative analysis. We consider 7 consecutive frames, divided into 2 overlapping groups of 4 frames each.

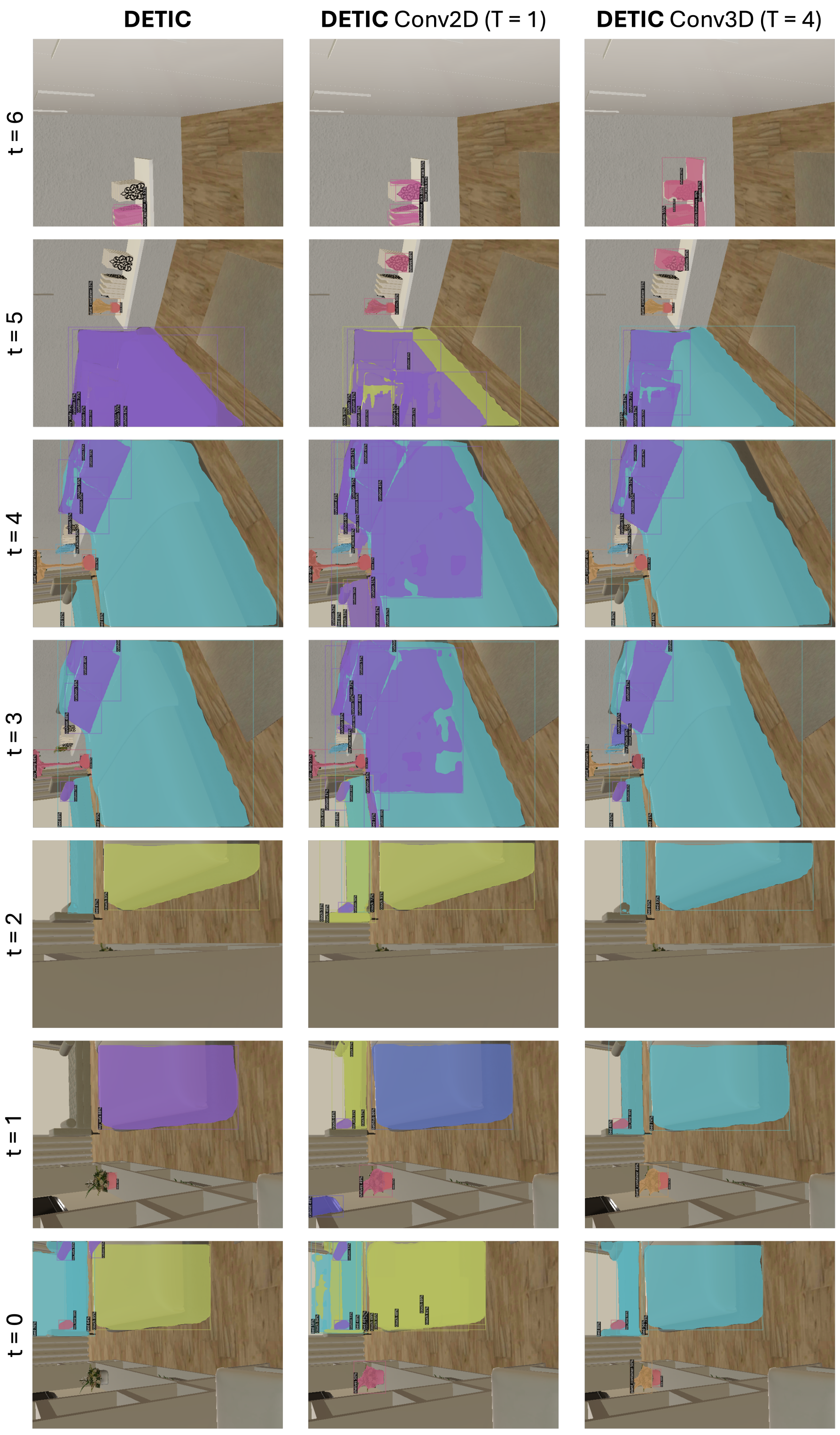


Figure 5.3: Comparative visualization of DETIC model configurations. DETIC with Conv3D $T = 4$ shows more consistency in segmentation mask colors across frames, indicating the model's capability to maintain stable predictions over time.

In our analysis presented in the previous figure, we compare the performance of the DETIC model, DETIC with 2DConv adapters ($T = 1$), and DETIC with 3DConv adapters ($T = 4$). In particular, the scene under consideration features a bed in the foreground and a couch in the background, providing a basis for evaluating the models' detection and consistency capabilities.

- DETIC with 2DConv adapters ($T = 1$) tends to make more dense predictions for the same object, notably the couch, producing different masks. This phenomenon is attributed to the spatial adaptation capabilities of the 2DConv adapters, which, while enhancing object delineation, may not contribute to temporal consistency.
- DETIC with 3DConv adapters ($T = 4$), on the other hand, demonstrates an improved ability to correctly predict the bed in the foreground as opposed to the couch, likely due to leveraging temporal information from previous frames (not shown here) where the model had already accurately identified the bed. This underscores the effectiveness of 3DConv adapters in enhancing temporal consistency across frames.
- However, the same level of accuracy is not observed for the couch in the background with DETIC with 3DConv adapters ($T = 4$), as it erroneously predicts the couch as a bed but maintains this prediction across frames. In this study, the primary focus is on consistency rather than outright correctness, and in this regard, DETIC with 3DConv adapters ($T = 4$) exhibits the desired trait of preserving its predictions over time.
- Similar observations are made for the plant container in the background, which remains consistently identified unlike the results from DETIC with 2DConv adapters ($T = 1$). This trend also applies to the shelves object, illustrating that when the robot turns or goes forward, DETIC with 3DConv adapters ($T = 4$) is more likely to maintain consistent identification of objects across the scene.

5.4 Success Rate Analysis

In the final evaluation, we conducted simulations in the HomeRobot environment to measure the performance of both the original model and the adapted model equipped with Spatio-Temporal Adapters. These simulations were carried out across 60 episodes, focusing on the 3 **out-of-domain scenes** that were not seen during the training phase. Each episode was capped at a maximum of 1250 steps, ensuring that the simulation would automatically conclude after reaching this limit, regardless of the task's completion status.

Both models operated under the same environment and policy, differing only in the object detection model utilized. This approach allowed us to systematically compare the effectiveness of the original DETIC model against the modified version with adapters in navigating and performing object manipulation tasks within unfamiliar environments. Through this comparative analysis, we aimed to identify any performance improvements or declines, in the context of egocentric object detection and temporal consistency.

Simulation Metric	**Original Model**	**Model with Adapters** `(T = 1)`	**Model with Adapters** `(T = 4)`	**Ground Truth**
`num_steps`	950	985	1050	705
`find_object_phase_success`	9.99%	11.66%	13.33%	62.7%
`pick_object_phase_success`	3.33%	3.33%	3.33%	29.7%
`find_recep_phase_success`	0.0%	0.0%	0.0%	43.3%
`partial_success`	3.33%	3.74%	4.16%	33.92%

Table 5.3: Comparison of simulation metrics between different agent perception configurations.

Table 5.3 provides a detailed comparison of simulation metrics across different configurations of agent perception models, contrasting the impact of incorporating adapters with temporal settings ($T = 1$ and $T = 4$) against the original model, with reference to ground truth values for context. In this analysis, the agent operates under an heuristic policy, as detailed in Section 2.4.4. For the ground truth scenario, the agent functions without the perception module, instead directly receiving ground truth masks to inform the heuristic policy. This method serves to delineate the perception module's contributions from the agent's inherent performance capabilities.

The findings for each metric are explained as follows:

- **Number of Steps (`num_steps`)**: All models required more steps than the ground truth (705 steps), suggesting an area for improvement. The addition of temporal information appears to prompt a more cautious navigation approach, as seen in the increased steps for the $T = 1$ and $T = 4$ models compared to the original.

- **Find Object Phase Success (`find_object_phase_success`)**: An improvement in object detection is noted with the use of temporal adapters, especially with $T = 4$ showing the highest success rate. However, a considerable gap remains compared to the ground truth success rate of 62.7%, indicating the challenge in meeting ground truth performance with current detection capabilities.
- **Pick Object Phase Success (`pick_object_phase_success`)**: This metric measures the success rate of the agent's ability to pick up an object. In HomeRobot, this step is a high-level action rather than a policy. Given this approach, it is expected that the adaptation of temporal information via adapters does not yield improvements in this phase. The action's success is influenced more on the accuracy of object detection itself and previous agent positioning rather than the temporal consistency of object identification.
- **Find Receptacle Phase Success (`find_recep_phase_success`)**: The success rate for identifying a suitable receptacle remained unchanged across all model configurations, being never accomplished. An explanation for this outcome is that the episodes likely concluded before the agent could successfully locate a goal receptacle. This early termination of episodes suggests a need for further optimization in earlier phases of the task sequence to allow the agent more opportunity to demonstrate improved performance in locating receptacles.

Chapter 6

Conclusions

In conclusion, this thesis tackled the challenge of improving temporal consistency in egocentric object detection for robots operating in home environments. By integrating Spatio-Temporal Adapters (ST-Adapters) into the DETIC model, we aimed to enhance the model's ability to consistently recognize objects across consecutive frames, crucial for the Open-Vocabulary Mobile Manipulation (OVMM) challenge within the HomeRobot 3D simulation environment.

Our experiments, utilizing the Habitat Synthetic Scenes Dataset (HSSD), revealed that adding temporal information processing to DETIC significantly reduced "noisy" predictions, leading to more reliable object detection. Specifically, the model with $T = 4$ showed improvements in the `find_object_phase_success` rate, demonstrating the added value of incorporating temporal data.

Despite these advancements, there is a scope for further development. Future efforts should concentrate on refining these models, with a particular emphasis on enhancing temporal reasoning capabilities. A critical area for future work involves moving beyond the heuristic agent used in our experiments by evaluating the reinforcement learning baseline of HomeRobot. This shift could potentially yield more dynamic and adaptive behaviors in complex environments. Additionally, testing with a more extended history (more frames) should be pursued; the current limitation to (T = 4) frame values was primarily due to computational constraints. Lastly, transitioning from simulated datasets to real-world datasets and environments represents an important step for future developments. While simulation provides a controlled setting for initial testing and development, incorporating real-world datasets is crucial for evaluating the model's performance under different conditions.

Overall, this work aimed to contribute to the field by testing a method to improve temporal consistency in robotic vision systems, setting the stage for further research in creating more reliable and context-aware robotic assistants for complex domestic environments.

Appendix A

Appendix

A.1 OVMM Classes List

In the context of the Open-Vocabulary Mobile Manipulation (OVMM) tasks, the agent is designed to recognize and interact with a predefined set of 150 classes. These classes include a variety of receptacles and objects, selected to encompass a broad spectrum of common household items. Below is the comprehensive list of classes that the OVMM agent is focusing to recognize and manipulate:

```
hssd_thing_classes = classes = [
"action_figure",
"android_figure",
"apple",
"backpack",
"baseballbat",
"basket",
"basketball",
"bath_towel",
"battery_charger",
"board_game",
"book",
"bottle",
"bowl",
"box",
"bread",
"bundt_pan",
"butter_dish",
"c-clamp",
"cake_pan",
"can",
"can_opener",
"candle",
    "candle_holder",
    "candy_bar",
    "canister",
    "carrying_case",
    "casserole",
    "cellphone",
    "clock",
    "cloth",
    "credit_card",
    "cup",
    "cushion",
    "dish",
    "doll",
    "dumbbell",
    "egg",
    "electric_kettle",
    "electronic_cable",
    "file_sorter",
    "folder",
    "fork",
    "gaming_console",
    "glass",
    "hammer",
    "hand_towel",
    "handbag",
    "hard_drive",
    "hat",
    "helmet",
    "jar",
    "jug",
    "kettle",
    "keychain",
    "knife",
    "ladle",
    "lamp",
    "laptop",
    "laptop_cover",
    "laptop_stand",
    "lettuce",
    "lunch_box",
    "milk_frother_cup",
    "monitor_stand",
    "mouse_pad",
    "multiport_hub",
    "newspaper",
    "pan",
    "pen",
    "pencil_case",
    "phone_stand",
    "picture_frame",
    "pitcher",
    "plant_container",
    "plant_saucer",
    "plate",
    "plunger",
    "pot",
    "potato",
    "ramekin",
    "remote",
    "salt_and_pepper_shaker",
    "scissors",
    "screwdriver",
    "shoe",
    "soap",
    "soap_dish",
    "soap_dispenser",
    "spatula",
    "spectacles",
    "spicemill",
    "sponge",
    "spoon",
    "spray_bottle",
    "squeezer",
    "statue",
    "stuffed_toy",
    "sushi_mat",
    "tape",
    "teapot",
    "tennis_racquet",
    "tissue_box",
    "toiletry",
    "tomato",
    "toy_airplane",
    "toy_animal",
    "toy_bee",
    "toy_cactus",
    "toy_construction_set",
    "toy_fire_truck",
    "toy_food",
    "toy_fruits",
    "toy_lamp",
    "toy_pineapple",
    "toy_rattle",
    "toy_refrigerator",
    "toy_sink",
    "toy_sofa",
    "toy_swing",
    "toy_table",
    "toy_vehicle",
    "tray",
    "utensil_holder_cup",
    "vase",
    "video_game_cartridge",
    "watch",
    "watering_can",
    "wine_bottle",
    "bathtub",
    "bed",
    "bench",
    "cabinet",
    "chair",
    "chest_of_drawers",
    "couch",
    "counter",
    "filing_cabinet",
    "hamper",
    "serving_cart",
    "shelves",
    "shoe_rack",
    "sink",
    "stand",
    "stool",
    "table",
    "toilet",
    "trunk",
    "wardrobe",
    "washer_dryer",
]
```

Bibliography

[1] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. «ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects». In: *CoRR* abs/2006.13171 (2020). arXiv: `2006.13171`. URL: `https://arxiv.org/abs/2006.13171` (cit. on p. 1).

[2] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. *Navigating to Objects in the Real World*. 2022. arXiv: `2212.00922 [cs.RO]` (cit. on pp. 1, 18).

[3] Thomas Wisspeintner, Tijn Zant, Luca Iocchi, and Stefan Schiffer. «RoboCup@Home: Scientific Competition and Benchmarking for Domestic Service Robots». In: *Interaction Studies* 10 (Dec. 2009), pp. 392–426. DOI: `10.1075/is.10.3.06wis` (cit. on pp. 1, 9).

[4] «Experiences with an interactive museum tour-guide robot». In: *Artificial Intelligence* 114.1 (1999), pp. 3–55. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/S0004-3702(99)00070-3`. URL: `https://www.sciencedirect.com/science/article/pii/S0004370299000703` (cit. on p. 1).

[5] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. *ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks*. 2020. arXiv: `1912.01734 [cs.CV]` (cit. on p. 1).

[6] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. *VirtualHome: Simulating Household Activities via Programs*. 2018. arXiv: `1806.07011 [cs.CV]` (cit. on p. 1).

[7] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: `2103.00020 [cs.CV]` (cit. on pp. 2, 34).

[8] Will Kay et al. *The Kinetics Human Action Video Dataset*. 2017. arXiv: `1705.06950 [cs.CV]` (cit. on p. 6).

[9] Chunhui Gu et al. *AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions*. 2018. arXiv: `1705.08421 [cs.CV]` (cit. on p. 6).

[10] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild.* 2012. arXiv: `1212.0402 [cs.CV]` (cit. on p. 6).

[11] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. «ActivityNet: A large-scale video benchmark for human activity understanding». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2015, pp. 961–970. DOI: `10.1109/CVPR.2015.7298698` (cit. on p. 6).

[12] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. *HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips.* 2019. arXiv: `1906.03327 [cs.CV]` (cit. on p. 6).

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848` (cit. on p. 6).

[14] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context.* 2015. arXiv: `1405.0312 [cs.CV]` (cit. on p. 6).

[15] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. *Open-vocabulary Object Detection via Vision and Language Knowledge Distillation.* 2022. arXiv: `2104.13921 [cs.CV]` (cit. on pp. 8, 25, 32).

[16] L. D. Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. «The DARPA LAGR program: Goals, challenges, methodology, and phase I results». In: *Journal of Field Robotics* 23.11-12 (2006), pp. 945–973. DOI: `https://doi.org/10.1002/rob.20161`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20161`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20161` (cit. on p. 9).

[17] Matthias Müller and Vladlen Koltun. *OpenBot: Turning Smartphones into Robots.* 2021. arXiv: `2008.10631 [cs.RO]` (cit. on p. 9).

[18] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. *Open-vocabulary Queryable Scene Representations for Real World Planning.* 2022. arXiv: `2209.09874 [cs.RO]` (cit. on p. 9).

[19] Krishna Murthy Jatavallabhula et al. *ConceptFusion: Open-set Multimodal 3D Mapping.* 2023. arXiv: `2302.07241 [cs.CV]` (cit. on p. 9).

[20] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. *CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory.* 2023. arXiv: `2210.05663 [cs.RO]` (cit. on p. 9).

[21] Benjamin Bolte, Austin Wang, Jimmy Yang, Mustafa Mukadam, Mrinal Kalakrishnan, and Chris Paxton. *USA-Net: Unified Semantic and Affordance Representations for Robot Memory.* 2023. arXiv: `2304.12164 [cs.RO]` (cit. on p. 9).

[22] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. *Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments.* 2020. arXiv: `2004.02857 [cs.CV]` (cit. on p. 9).

[23] Weiyu Liu, Yilun Du, Tucker Hermans, Sonia Chernova, and Chris Paxton. *StructDiffusion: Language-Guided Creation of Physically-Valid Structures using Unseen Objects.* 2023. arXiv: `2211.04604 [cs.RO]` (cit. on p. 9).

[24] Austin Stone et al. *Open-World Object Manipulation using Pre-trained Vision-Language Models.* 2023. arXiv: `2303.00905 [cs.RO]` (cit. on p. 9).

[25] Danny Driess et al. *PaLM-E: An Embodied Multimodal Language Model.* 2023. arXiv: `2303.03378 [cs.LG]` (cit. on p. 9).

[26] Mukul Khanna et al. *Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation.* 2023. arXiv: `2306.11290 [cs.CV]` (cit. on p. 10).

[27] Manolis Savva et al. *Habitat: A Platform for Embodied AI Research.* 2019. arXiv: `1904.01201 [cs.CV]` (cit. on p. 10).

[28] Andrew Szot et al. *Habitat 2.0: Training Home Assistants to Rearrange their Habitat.* 2022. arXiv: `2106.14405 [cs.LG]` (cit. on pp. 10, 18, 19).

[29] Charles C. Kemp, Aaron Edsinger, Henry M. Clever, and Blaine Matulevich. *The Design of Stretch: A Compact, Lightweight Mobile Manipulator for Indoor Human Environments.* 2022. arXiv: `2109.10892 [cs.RO]` (cit. on p. 10).

[30] Sriram Yenamandra et al. *HomeRobot: Open-Vocabulary Mobile Manipulation.* 2024. arXiv: `2306.11565 [cs.RO]` (cit. on pp. 10, 14, 15, 19, 21).

[31] Andrew Melnik, Michael Büttner, Leon Harz, Lyon Brown, Gora Chand Nandi, Arjun PS, Gaurav Kumar Yadav, Rahul Kala, and Robert Haschke. *UniTeam: Open Vocabulary Mobile Manipulation Challenge.* 2023. arXiv: `2312.08611 [cs.RO]` (cit. on p. 18).

[32] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. *Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments.* 2018. arXiv: `1711.07280 [cs.CV]` (cit. on p. 18).

[33] Dhruv Batra et al. *Rearrangement: A Challenge for Embodied AI.* 2020. arXiv: `2011.01975 [cs.AI]` (cit. on p. 18).

[34] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. *Object Goal Navigation using Goal-Oriented Semantic Exploration*. 2020. arXiv: 2007.00643 [cs.CV] (cit. on p. 18).

[35] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. *Visual Room Rearrangement*. 2021. arXiv: 2103.16544 [cs.CV] (cit. on p. 18).

[36] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. «Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?» In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 6670–6677. ISSN: 2377-3774. DOI: 10.1109/lra.2020.3013848. URL: http://dx.doi.org/10.1109/LRA.2020.3013848 (cit. on p. 18).

[37] Joanne Truong, Sonia Chernova, and Dhruv Batra. «Bi-Directional Domain Adaptation for Sim2Real Transfer of Embodied Navigation Agents». In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2634–2641. DOI: 10.1109/LRA.2021.3062303 (cit. on p. 18).

[38] Joanne Truong, Max Rudolph, Naoki Yokoyama, Sonia Chernova, Dhruv Batra, and Akshara Rai. *Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation*. 2022. arXiv: 2207.10821 [cs.RO] (cit. on p. 18).

[39] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. *Matterport3D: Learning from RGB-D Data in Indoor Environments*. 2017. arXiv: 1709.06158 [cs.CV] (cit. on pp. 18, 19).

[40] Santhosh K. Ramakrishnan et al. *Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI*. 2021. arXiv: 2109.08238 [cs.CV] (cit. on pp. 18, 19).

[41] Fei Xia, Amir Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. *Gibson Env: Real-World Perception for Embodied Agents*. 2018. arXiv: 1808.10654 [cs.AI] (cit. on p. 18).

[42] Karmesh Yadav et al. *Habitat-Matterport 3D Semantics Dataset*. 2023. arXiv: 2210.05633 [cs.CV] (cit. on p. 18).

[43] Matt Deitke et al. *ProcTHOR: Large-Scale Embodied AI Using Procedural Generation*. 2022. arXiv: 2206.06994 [cs.AI] (cit. on pp. 18, 19).

[44] Eric Kolve et al. *AI2-THOR: An Interactive 3D Environment for Visual AI*. 2022. arXiv: 1712.05474 [cs.CV] (cit. on p. 18).

[45] Agrim Gupta, Piotr Dollár, and Ross Girshick. *LVIS: A Dataset for Large Vocabulary Instance Segmentation.* 2019. arXiv: 1908.03195 [cs.CV] (cit. on p. 23).

[46] Alina Kuznetsova et al. «The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale». In: *International Journal of Computer Vision* 128.7 (Mar. 2020), pp. 1956–1981. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01316-z. URL: http://dx.doi.org/10.1007/s11263-020-01316-z (cit. on p. 23).

[47] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 23).

[48] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. *Detecting Twenty-thousand Classes using Image-level Supervision.* 2022. arXiv: 2201.02605 [cs.CV] (cit. on pp. 23, 24).

[49] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey Chiang, Zhihao Wu, and Xiaowei Ding. «Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation». In: *Medical Image Analysis* 63 (Apr. 2020), p. 101693. DOI: 10.1016/j.media.2020.101693 (cit. on p. 24).

[50] Shijie Fang, Yuhang Cao, Xinjiang Wang, Kai Chen, Dahua Lin, and Wayne Zhang. *WSSOD: A New Pipeline for Weakly- and Semi-Supervised Object Detection.* 2021. arXiv: 2105.11293 [cs.CV] (cit. on p. 25).

[51] Zeyi Huang, Yang Zou, Vijayakumar Bhagavatula, and Dong Huang. *Comprehensive Attention Self-Distillation for Weakly-Supervised Object Detection.* 2020. arXiv: 2010.12023 [cs.CV] (cit. on p. 25).

[52] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. *Unbiased Teacher for Semi-Supervised Object Detection.* 2021. arXiv: 2102.09480 [cs.CV] (cit. on p. 25).

[53] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. *PCL: Proposal Cluster Learning for Weakly Supervised Object Detection.* 2018. arXiv: 1807.03342 [cs.CV] (cit. on p. 25).

[54] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. «Open-Vocabulary Object Detection Using Captions». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2021, pp. 14393–14402 (cit. on p. 25).

[55] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. *Scaling Open-Vocabulary Image Segmentation with Image-Level Labels.* 2022. arXiv: 2112.12143 [cs.CV] (cit. on p. 25).

[56] Boyi Li, Kilian Q. Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. *Language-driven Semantic Segmentation*. 2022. arXiv: `2201.03546 [cs.CV]` (cit. on p. 25).

[57] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: `2103.14030 [cs.CV]` (cit. on pp. 26, 28, 30).

[58] Can Cui, Linhan Xu, Boyu Yang, and Jun Ke. «Meta-TR: Meta-Attention Spatial Compressive Imaging Network With Swin Transformer». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (Jan. 2022), pp. 1–12. DOI: `10.1109/JSTARS.2022.3194949` (cit. on p. 30).

[59] Rishi Bommasani et al. *On the Opportunities and Risks of Foundation Models*. 2022. arXiv: `2108.07258 [cs.LG]` (cit. on p. 34).

[60] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: `1810.04805 [cs.CL]` (cit. on p. 34).

[61] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: `2005.14165 [cs.CL]` (cit. on p. 34).

[62] Alec Radford and Karthik Narasimhan. «Improving Language Understanding by Generative Pre-Training». In: 2018. URL: `https://api.semanticscholar.org/CorpusID:49313245` (cit. on p. 34).

[63] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. *Parameter-Efficient Transfer Learning for NLP*. 2019. arXiv: `1902.00751 [cs.LG]` (cit. on p. 34).

[64] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. «AdapterHub: A Framework for Adapting Transformers». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 46–54. URL: `https://www.aclweb.org/anthology/2020.emnlp-demos.7` (cit. on p. 34).

[65] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. *VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding*. 2021. arXiv: `2109.14084 [cs.CV]` (cit. on p. 34).

[66] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. *Slow-Fast Networks for Video Recognition.* 2019. arXiv: 1812.03982 [cs.CV] (cit. on pp. 34, 35).

[67] Joao Carreira and Andrew Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset.* 2018. arXiv: 1705.07750 [cs.CV] (cit. on p. 34).

[68] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. «Learning spatiotemporal features with 3d convolutional networks». In: *IEEE CVPR.* 2015, pp. 4489–4497 (cit. on p. 34).

[69] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. *Temporal Segment Networks: Towards Good Practices for Deep Action Recognition.* 2016. arXiv: 1608.00859 [cs.CV] (cit. on p. 34).

[70] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. «Is Space-Time Attention All You Need for Video Understanding?» In: *International Conference on Machine Learning.* PMLR. 2021, pp. 813–824 (cit. on p. 35).

[71] Yun Liu, Yu-Huan Wu, Guolei Sun, Le Zhang, Ajad Chhatkuli, and Luc Van Gool. *Vision Transformers with Hierarchical Attention.* 2024. arXiv: 2106.03180 [cs.CV] (cit. on p. 35).

[72] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. *ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning.* 2022. arXiv: 2206.13559 [cs.CV] (cit. on pp. 35, 36, 42).

[73] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. *ViViT: A Video Vision Transformer.* 2021. arXiv: 2103.15691 [cs.CV] (cit. on p. 35).

[74] Hao Zhou, Tiancheng Shen, Xu Yang, Hai Huang, Xiangtai Li, Lu Qi, and Ming-Hsuan Yang. *Rethinking Evaluation Metrics of Open-Vocabulary Segmentaion.* 2023. arXiv: 2311.03352 [cs.CV] (cit. on p. 47).

[75] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2.* https://github.com/facebookresearch/detectron2. 2019 (cit. on p. 53).