

# Representation Learning for Active Perception

*Jerome Quenum*



Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/EECS-2025-153

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-153.html>

August 12, 2025

Copyright © 2025, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.



Representation Learning for Active Perception

by

Jerome Quenum

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Co-chair

Professor Jitendra Malik, Co-chair

Professor Alexei Efros

Dr. Mark Lowell

Summer 2025

# Representation Learning for Active Perception

Copyright 2025  
by  
Jerome Quenum

# Abstract

## Representation Learning for Active Perception

by

Jerome Quenum

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Trevor Darrell, Co-chair

Professor Jitendra Malik, Co-chair

Active perception driven by representation learning lies at the intersection of computer vision and robotics, powered by advanced Artificial Intelligence (AI) algorithms. It has for goal to develop systems that can actively gather information from their environment by learning useful representations to enhance both their perception and decision-making capabilities. This dissertation explores various facets of this topic, with a particular focus on how advanced representation-learning techniques can significantly improve the performance and utility of such systems.

We begin by examining the role of precision in AI-driven systems through classical segmentation tasks in Ultra-High-Resolution (UHR) images within the context of battery design. A novel transformer-based network, TransforCNN, is introduced to segment dendrites in Lithium Metal Battery (LMB) 3D X-ray computed tomography (XCT) volumes. This approach significantly improves the accuracy of segmenting critical structures, which is essential for developing more efficient and reliable batteries. The precision gained in this segmentation problem through our proposed representation learning model not only enhances the visual understanding of battery components but also lays the groundwork for more informed and effective design strategies.

Building on this foundation, we transition to addressing the challenge of processing Ultra-High-Resolution (UHR) images, which is important for applications requiring both speed and accuracy. We explore this in the context of barcode detection in UHR images, where a new detection pipeline is proposed, integrating a modified Region Proposal Network (RPN) with a proposed segmentation network named Y-Net. This study demonstrates how representation learning can be optimized to reduce latency while maintaining high accuracy, showcasing the potential of AI to handle complex visual tasks in real time.

We then examine scalability when it comes to the integration of representation learning for active perception in robotics, where speed, accuracy, adaptability, and robustness are key. Using the Open X Embodiment (OXE) dataset, we investigate a cross-robot self-supervised sensorimotor pre-training approach through RPTx, a multimodal model that does not leverage language instructions and does not use auto-regressive methods. This approach provided valuable insights into the robustness and adaptability issues encountered by robotic systems. Here, we conduct reliability assessments, an examination of negative transfer instances where models struggle to adapt, and we speculate on future directions for overcoming these difficulties. Additionally, we observe that the use of multimodal models that leverage large language models for instruction tuning and that employ auto-regressive techniques led to more stable and promising outcomes with LLARVA, a vision-action instruction tuning paradigm that enhances Robot Learning. These findings demonstrate how learning good representations can enhance robotic systems’ ability to interpret and respond to complex environmental cues.

Finally, we explore the challenge of extending segmentation models beyond fixed object categories to support reasoning over visual scenes. While current segmentation models work well when objects are clearly defined, they struggle with complex user queries that refer to multiple or implicit objects. Recent work in reasoning segmentation, which generates segmentation masks from natural language input, shows that vision-language models (VLMs) can help address this. However, our experiments show that existing models perform poorly on remote-sensing images, which often contain dense and varied content. To address this, we introduce LISAT, a vision-language model designed to describe remote-sensing images, answer questions about them, and segment objects based on user queries. LISAT is trained on GRES, a dataset with 27,615 annotations across 9,205 images, and PreGRES, a multimodal dataset with over one million question–answer pairs. LISAT outperforms geospatial models like RS-GPT4V by 10.04% in BLEU-4 for image description and improves on open-domain models in reasoning segmentation by 143.36% in generalized Intersection over Union (gIoU). These results show how learning strong multimodal representations can improve scene understanding, especially in the geospatial domain.

In memory of my parents,  
Fatouma and Emile.

In memory of my sister,  
Yolande.

To my sisters,  
Marleine, Edwige, and Ghislaine.

# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Thesis outline . . . . .	3
<b>2 Precision: Lithium Metal Battery Quality Control via Transformer-CNN Segmentation</b>	<b>4</b>
2.1 Introduction . . . . .	4
Assessing Battery Quality with Imaging . . . . .	4
Deep Learning for Semantic Segmentation . . . . .	5
Problem and Motivation . . . . .	7
Research Contributions . . . . .	8
2.2 Materials Description . . . . .	8
Electrochemical Testing . . . . .	10
Synchrotron X-ray CT Imaging . . . . .	10
Raw data Pre-Processing . . . . .	11
2.3 Computational Methods . . . . .	12
U-Net . . . . .	13
Y-Net . . . . .	14
TransforCNN . . . . .	14
E-Net . . . . .	15
2.4 Experimental Results . . . . .	16
2.5 Discussion . . . . .	18
<b>3 Latency: Fast, Accurate Barcode Detection in Ultra-High-Resolution Images</b>	<b>23</b>
3.1 Introduction . . . . .	23

3.2	Proposed Approach . . . . .	26
	Modified Region Proposal Network . . . . .	26
	Y-Net Segmentation Network . . . . .	27
	Bounding Box Extraction . . . . .	29
3.3	Datasets and Results . . . . .	30
3.4	Discussion . . . . .	32
<b>4</b>	<b>Active Perception: Using Non-Auto-Regressive Multimodal Models without Language Instructions via RPTx, a Cross Robot Learning Method with Sensorimotor Pre-training</b>	<b>33</b>
4.1	Introduction . . . . .	33
4.2	Cross Robot Learning with Sensorimotor Pre-training . . . . .	34
4.3	RPTx . . . . .	36
4.4	Downstream Transfer . . . . .	37
4.5	Discussion . . . . .	37
	Negative Transfer Assessment . . . . .	37
	Reliability Assessment . . . . .	43
	Future Directions and Speculation . . . . .	48
<b>5</b>	<b>Active Perception: Using Auto-Regressive Multimodal Models with Language Instructions via LLARVA, a Vision-Action Instruction Tuning Method that Enhances Robot Learning</b>	<b>50</b>
5.1	Introduction . . . . .	50
5.2	LLARVA . . . . .	51
5.3	Method . . . . .	52
5.4	Discussion . . . . .	56
5.5	Future Work . . . . .	57
<b>6</b>	<b>Active Perception: Using Auto-Regressive Multimodal Models for Geospatial Reasoning Segmentation via LISat, a Language-Instructed Segmentation Assistant for Satellite Imagery</b>	<b>58</b>
6.1	Introduction . . . . .	58
6.2	Related Work . . . . .	60
	Remote-Sensing Datasets for Multimodal Learning . . . . .	60
	Geospatial Foundation Models . . . . .	61
	Reasoning Segmentation . . . . .	62
6.3	Geospatial Reasoning Segmentation Dataset . . . . .	62
	PreGRES . . . . .	62
	GRES . . . . .	63
6.4	Training LISAT for Geospatial Reasoning Segmentation . . . . .	65
	Geospatial Multimodal Language Models . . . . .	65
	Preliminaries . . . . .	65

Training Objectives . . . . .	66
6.5 Experimental Results . . . . .	67
Segmentation . . . . .	68
Captioning and Question-Answering . . . . .	70
Limitations and Failure Cases . . . . .	72
6.6 Discussion . . . . .	73
<b>7 Conclusion</b>	<b>75</b>
<b>Bibliography</b>	<b>77</b>
<b>A Extension of Battery Operation</b>	<b>99</b>
A.1 Overview of Lithium-Ion and Lithium-Metal Batteries . . . . .	99
A.2 How Lithium-Ion Batteries Work . . . . .	100
A.3 How Lithium Metal Batteries Work . . . . .	102
A.4 Why Move from Lithium-Ion Batteries to Lithium Metal Batteries . . . . .	103
A.5 Dendrite Formation and its Risks . . . . .	103
<b>B Extension of Y-Net</b>	<b>105</b>
B.1 Y-Net Robustness and Generalization . . . . .	105
B.2 Y-Net for Depth and Uncertainty Estimation . . . . .	106
<b>C Training Multimodal Models on High Performance Computers (HPC)</b>	<b>110</b>
C.1 Code Release . . . . .	110
C.2 Manual Launch Script . . . . .	111
C.3 Automatic Launch Script . . . . .	112
C.4 DeepSpeed Launch Script . . . . .	115
C.5 More on OXE and LLARVA Datasets . . . . .	116
<b>D Extension of LISAt</b>	<b>119</b>
D.1 Code Release . . . . .	119
D.2 More on GRES . . . . .	119
Prompt Engineering . . . . .	119
Dataset Quality Assurance . . . . .	122
GRES Dataset Summary . . . . .	122
Additional Experiments . . . . .	123
D.3 More on PreGRES . . . . .	129
D.4 Qualitative Analysis . . . . .	129
Success Cases of LISAT . . . . .	131
Failure Cases of LISAT . . . . .	131
Ground Truth Error Cases . . . . .	131



# List of Figures

2.1	Types of Hybrid architectures. . . . .	6
2.2	Diagram illustrating the Li-polymer-Li symmetric cell design, imaged using X-ray CT, with highlighted dendrite formations (blue) and the redeposited Li (red). . . . .	7
2.3	Cross-Sectional Images for the Li-polymer-Li symmetric cell; (A) Cross section of the x-y plane where the training was done on this plane; (B) Cross-sections of the x-z plane and detailing of the cell components; (C) Cross-sections of the y-z plane. . . . .	9
2.4	Schematic illustration of the pouch cell. . . . .	11
2.5	Schematic illustration of the dataset collection process. This process was outlined in (1) . . . . .	11
2.6	Sample Raw data obtained after TomoPy reconstruction of a CT scan . . . . .	12
2.7	Sample Region of Interest (RoI) data obtained after pre-processing TomoPy reconstruction of a CT scan . . . . .	13
2.8	U-Net architecture adapted from (2). . . . .	13
2.9	T-Net architecture. . . . .	15
2.10	Training curves for U-Net, Y-Net, and TransforCNN on the y-axis vs numbers of epochs on the x-axis; the models were optimized over the binary cross-entropy function as loss and evaluated on the dice similarity coefficient as evaluation metric during training; The gray curves depict behavior on the validation sets while the black curves show behavior on the training sets over increasing numbers of epochs; (a) training and validation dice coefficient for U-Net; (b) training and validation dice coefficient for Y-Net; (c) training and validation Dice coefficient for TransforCNN; (d) training and validation loss for U-Net; (e) training and validation loss for Y-Net; (f) training and validation loss for U-Net; the use of dropout during only the training phase explains why the models tend to perform better on the validation set over time. . . . .	17
2.11	U-Net, Y-Net, T-Net, and E-Net sample outputs on a slice showing a cross-section along the x-y plane. . . . .	20
2.12	U-Net, Y-Net, T-Net, and E-Net sample outputs on random test patches. . . . .	21
2.13	3D rendering of U-Net, Y-Net, T-Net, and E-Net on test volume; (a) grayscale test input volume; (b), (c), (d), (e) are respectively U-Net, Y-Net, T-Net, and E-Net predictions . . . . .	22

3.1	Barcode detection with classical Signal Processing. . . . .	23
3.2	Barcode detection in UHR images. . . . .	24
3.3	Proposed approach, the modified RPN is followed by Y-Net and the bounding box extractor. . . . .	26
3.4	Sample region proposal processing phases; (a) UHR Input image; (b) a Low Resolution (LR) of the input is obtained; (c) proposed regions on (b); (d) remapped regions from (b) to (a). . . . .	27
3.5	Sample outputs of our pipeline; yellow - segmented barcode pixels; purple - segmented background pixels; boxes - bounding box extracted; (a) synthetic barcode image; (b) real barcode image; (c) prediction results on (a); (d) prediction results on (b). . . . .	28
3.6	Y-Net architecture. . . . .	29
3.7	(a) Y-Net output; (b) Y-Net output after erosion; (c) extracted bounding boxes –red, ground truth bounding boxes –green on eroded output; (d) final bounding boxes after pixel correction margin on Y-Net output; (e) Y-Net output of occluded barcodes scenarios; (f) final extracted bounding boxes are grouped after pixel correction margin due to overlapping barcodes in the input image. . . . .	32
4.1	Architecture of RPT (3). . . . .	35
4.2	Modality Distribution across 54 OXE dataset; no data is missing all modality at once. . . . .	36
4.3	PCA of ViT features of the right camera. . . . .	38
4.4	Evaluation on the Picking task. . . . .	39
4.5	Evaluation on the Destack task. . . . .	39
4.6	Evaluation on the Stacking task. . . . .	40
4.7	Success rate vs. context length. . . . .	41
4.8	Success rate vs. pre-training data size. . . . .	42
4.9	Success rate vs. masking type. . . . .	43
4.10	Success rate vs. robot type (Cross Robot Transfer). . . . .	44
4.11	Success rate vs. consistency test over several runs. . . . .	45
5.1	Overview of LLARVA (4). We used language instruction that contains <i>robot model</i> , <i>control mode</i> , <i>robot task</i> , <i>proprioceptive information</i> , and <i>number of predicted steps</i> , and outputs text with the next <i>robot action(s)</i> and the <i>visual trace</i> for the remainder of the episode. . . . .	52
5.2	LLARVA architecture (4). . . . .	53

6.1	Existing models struggle to generate accurate segmentation masks for complex natural language queries in remote-sensing imagery. LISAT, our open-source, open-data, foundation model for geospatial reasoning segmentation trained on GRES, our new semi-synthetic dataset for remote-sensing reasoning segmentation, helps to bridge the gap between State-of-the-Art (SOTA) reasoning segmentation models and remote-sensing domains. . . . .	59
6.2	To generate synthetic data, we start with a seed detection dataset ( <b>xView</b> ). We then filter detections for those that are both visually interesting and highly distinguishable (A). For that detection, we then generate a natural language description (B) and a pixel-wise segmentation mask (C). Finally, the natural language description is used to generate a localization query (D). Together, the segmentation mask and the query form a ground-truth pair for the LISAT reasoning segmentation fine-tuning. . . . .	64
6.3	LISAT integrates a geospatial multimodal large language model (MLLM) with a segmentation decoder to enable reasoning-based segmentation. LISAT first pre-trains a Remote-CLIP-based MLLM on PreGRES before fine-tuning on GRES. We then expand the LMM vocabulary with a segmentation token (<SEG>), whose final-layer embedding is projected into the SAM segmentation query space and combined with image features to produce a segmentation mask. . . . .	66
6.4	Scaling behavior of LISAT on the GRES dataset. While adding additional data is helpful, even with <i>7K training</i> images (the full GRES dataset), we observe the beginning of a plateau in performance, particularly on cIOU scores. This suggests that more data alone may not be helpful, and instead, we may need additional data variance outside the <b>xView</b> classes. . . . .	70
A.1	Diagram of a battery at equilibrium. . . . .	100
A.2	Diagram of a battery during its discharging state. . . . .	101
A.3	Diagram of a battery during its charging state. . . . .	102
A.4	Dendrite formation in lithium metal batteries as shown in (5). . . . .	104
B.1	Y-Net segmentation predictions extending beyond incomplete ground truth annotations. . . . .	106
B.2	Qualitative results of Y-Net on depth and uncertainty estimation for the Stanford 2D-3D-S dataset. . . . .	108
B.3	Qualitative results of Y-Net on depth and uncertainty estimation for the Replica dataset. . . . .	109
C.1	Diagram of OXE dataset distribution as shown in (6); (a) shows the dataset distribution per robot; (b) shows that the Franka robot has the most diversity in distinct scenes; (c) shows that xArm and Google Robot have the most trajectories; (d, e) show the data distribution by skills and common objects. . . . .	116
C.2	Diagram of data distribution for LLARVA as shown in (4). . . . .	117

D.1	Class distribution of 33% training set . . . . .	124
D.2	Class distribution of 66% training set . . . . .	125
D.3	Class distribution of 100% training set . . . . .	126
D.4	Class distribution of validation set . . . . .	127
D.5	Class distribution of testing set . . . . .	128

# List of Tables

2.1	Comparison of Transformer encoder vs. CNN encoder . . . . .	6
2.2	Comparison of Li-ion and Li-metal Battery. . . . .	9
2.3	mIoU, mDSC, Inference Time, and Patch Size for Li-Li Symmetric battery Dataset. . . . .	18
2.4	Volume and Percentage Volume Occupied for U-Net, Y-Net, TransforCNN, and E-Net. . . . .	18
3.1	Average Precision for Max Detection of 100 and Average Recall for Max Detection of 10 computed using MS COCO API. . . . .	30
3.2	Mean IoU (mIoU), Precision, Recall, and Detection Rate (DR) at an IoU threshold of 0.5 for Muenster and ArTe-Lab datasets. . . . .	30
3.3	Pixel-wise metrics . . . . .	30
5.1	Success rate (%) of LLARVA on the Frank robot. We compare LLARVA with RPT, RPTx (480 trajectories), and Octo by taking each pre-trained model and fine-tuning it on the same set of demonstrations. LLARVA outperforms the others on all the tasks. . . . .	56
6.1	Qualitative examples of the segmentations generated by LISAT on the GRES dataset. . . . .	67
6.2	Performance of LISAT against LISA-7B-v1, LISA-13B-Llama2-v1, PixelLM-7B and PixelLM-13B on GRES across different object sizes. LISAT-7B consistently outperforms the baseline models, particularly in the Small object category. . . .	69
6.3	Comparison of LISAT’s performance using GeoSAM vs. SAM for segmentation on the A11 dataset configuration. . . . .	69
6.4	Comparison of captioning performance on the UCM-Captions dataset. Results are reported for BLEU-4 and CIDEr metrics. . . . .	71
6.5	Comparison of captioning performance on the NWPU-Captions dataset. Results are reported for BLEU-4 and SPICE metrics. . . . .	71
6.6	Performance on RSVQA-LR (% accuracy). . . . .	72
6.7	Ablations of the base language model and visual feature extractor for LISAT <sub>PRE</sub> on the UCM-Captions dataset. . . . .	73

C.1	More statistics about the LLARVA (4) dataset. . . . .	118
D.1	Summary of object categories across train, validation, and test sets . . . . .	123
D.2	Performance comparison across different query types on 10 test examples. . . . .	124
D.3	Performance comparison across different models and datasets. . . . .	125
D.4	Effect of input image Size on performance and inference time (All object sizes). Measured on a single NVIDIA A100. . . . .	126
D.5	Overview of data sources and statistics . . . . .	130
D.6	Comparison of various models for LISAT <sub>PRE</sub> on RSICD . . . . .	130
D.7	Comparison of various models for LISAT <sub>PRE</sub> on Sydney-Captions . . . . .	131
D.8	Comparison of vision and language encoders for LISAT <sub>PRE</sub> on UCM-Captions, NWPU-Captions, RSICD, and Sydney-Captions . . . . .	132
D.9	Performance metrics for LISAT <sub>PRE</sub> on the RSVQA_LR . . . . .	133
D.10	Comparison of predictions and ground truth across models . . . . .	134
D.11	Comparison of predictions and ground truth across models (Cont.) . . . . .	135
D.12	Comparison of predictions and ground truth across models (Cont.) . . . . .	136
D.13	Comparison of predictions and ground truth across models (Cont.) . . . . .	137
D.14	Comparison of predictions and ground truth across models (Cont.) . . . . .	138
D.15	Failure cases . . . . .	139
D.16	GT Mistake cases . . . . .	140

## Acknowledgments

First and foremost, I would like to thank my advisors, Profs. Trevor Darrell and Jitendra Malik. Their expertise, understanding, and patience have profoundly shaped my graduate experience. Their guidance and persistent support have been instrumental in reaching this point.

I am also grateful to my committee members, Prof. Alexei Efros and Dr. Mark Lowell, for their invaluable feedback and support throughout this process. Additionally, I extend my thanks to Prof. Anjgo Kanazawa and Dr. Daniela Ushizima for their endorsement and support.

I would like to thank my colleagues and friends, whose stimulating discussions and happy distractions provided much-needed relief from research. Your camaraderie and encouragement have made this journey far more enjoyable.

I am deeply grateful to my family for their unwavering love and support. In memory of my mother, Fatouma, and my father, Emile, whose belief in me and the values of perseverance and curiosity they instilled have guided me throughout this journey. In memory of my sister Yolande, whose spirit continues to inspire me. To my sisters Marleine, Edwige, and Ghislaine, thank you for always being there. Most of all, I would like to thank my sister Ghislaine for her unwavering support and love throughout my life, and for inspiring me to reach for my dreams.

Finally, I would like to acknowledge the Berkeley Artificial Intelligence Research (BAIR) Laboratory, the Lawrence Berkeley National Laboratory (LBNL), the U.S. Department of Defense (DoD), and the U.S. National Geospatial-Intelligence Agency (NGA) for their support during my studies. This work would not have been possible without their assistance.

This dissertation is dedicated to everyone who has supported me along this journey. Thank you for your encouragement and belief in me.

# Chapter 1

## Introduction

### 1.1 Motivation

Representation learning refers to the process of automatically discovering the features or representations needed for a task from raw data. Instead of manually crafting features (e.g., edges in images, patterns in text), representation learning allows models to learn these features directly from the data. Deep learning methods have been particularly very successful in learning such good representations. It is the case, for example, in image recognition where the network might learn hierarchical representations in a way that lower layers detect edges, middle layers detect shapes, and higher layers recognize objects (7; 8; 9).

Active perception, on the other hand is the idea that a perception system can take actions to improve its understanding of the environment. Unlike passive perception, where a system simply observes the environment, active perception involves actively choosing viewpoints, actions, or strategies to gather the most informative data. An example of this is in robotic systems where a robot would be moving around an object to see it from different angles, thereby gaining a better understanding of its shape and size (10).

Given the above, it is natural to deduce that being able to obtain good representations will significantly improve robotic systems, as those can allow the robots to acquire useful information about their environment, which will facilitate their ability to actively make decisions. In other words, we want learned representations to be as robust and informative as possible to guide the active perception process, regardless of environmental changes.

This dissertation addresses the multifaceted challenges of representation learning within the domain of active perception, exploring its impact across different applications. In the field of battery technology, achieving high precision in the segmentation of Lithium Metal Battery (LMB) structures is crucial for advancing design and efficiency. We discuss TransforCNN (T-Net) (11), a novel transformer-based network designed to enhance the segmentation accuracy of dendrites in 3D X-ray computed tomography (XCT) volumes. This approach not only shows that a system that can learn good representations improves visual understanding but also supports more effective design strategies in battery technology.



Shifting the focus to industrial applications, we tackle the challenge of processing ultra-high-resolution (UHR) images, where both speed and accuracy are critical. For tasks such as barcode detection, we discuss a detection pipeline that combines a modified Region Proposal Network (RPN) with a Y-Net segmentation network (12). This solution optimizes the learned representation of visual features to achieve real-time processing with minimal latency while maintaining high accuracy. This work highlights how advanced representation techniques can be tailored to meet the demands of real-time applications. For example, in Amazon warehouses, efficient barcode detection is necessary for managing inventory and streamlining operations. Similarly, barcode scanning in environments such as grocery stores and hospitals has been reported to enhance efficiency and minimize mistakes in daily operations.

In the realm of robotics, where systems must be adaptable and robust, we explore the integration of learning a good representation of sensory data with active perception. Some applications of this could be, for example, in gas station safety, where robots equipped with advanced perception capabilities can identify potential hazards and improve safety protocols. Another example is in self-driving vehicles, where robust perception and representation learning are critical for safe and efficient navigation. It is also the case in food delivery systems where robots with advanced perception and navigation technologies can efficiently handle complex delivery tasks and avoid obstacles. By employing the Open X Embodiment (OXE) dataset (6), we examine RPTx, an adaptation of RPT (3), a multimodal model that employs neither language nor auto-regressive techniques for robot self-supervised sensorimotor pre-training. This study provides insights into issues associated with robustness and adaptability when it comes to cross-robot learning. We present approaches for reliability assessment and analyze instances where models face challenges in new scenarios. Additionally, we discuss how using language and auto-regressive methods via the paradigm introduced in LLARVA contributes to stability and improved outcomes, paving the way for future advancements in active perception systems.

We further extend our investigation into how representation learning can support more open-ended visual tasks guided by user intent. In many real-world scenarios, particularly in geospatial analysis, users may seek information using flexible, language-based queries rather than relying on fixed object categories. This requires models that can effectively link visual content with natural language. To address this need, we introduce LISAT, a model trained on both imagery and text to perform tasks such as image description, question answering, and segmentation from natural language prompts. Using curated geospatial datasets, **PreGRES** and **GRES**, which include paired annotations and questions, LISAT shows how multimodal learning can aid decision-making in complex settings where goals and context may shift. This line of work highlights new opportunities for applying active perception in systems that must interpret user-defined objectives and respond in a context-aware manner.

Through these investigations, this dissertation aims to advance our understanding of how good representation learning can enhance AI systems' capabilities across various domains, ultimately improving their performance and adaptability in complex, real-world environments.

## 1.2 Thesis outline

This dissertation is structured into five main interconnected chapters, each addressing a unique aspect of representation learning for active perception, but all contributing to a cohesive exploration of the topic.

In chapter 2, we focus on the precision of AI-driven systems, particularly in the domain of battery design. Here, a novel transformer-based network, TransforCNN (T-Net), is introduced for segmenting dendrites in Lithium Metal Battery (LMB) 3D X-ray computed tomography (XCT) volumes. The purpose of this chapter is to demonstrate how a good representation learning scheme can significantly improve the accuracy of AI-driven systems. We also show how this segmentation model, can be used for the development of more efficient and reliable batteries, thereby giving insights into the foundation for more informed design strategies in energy storage technology.

While precision is essential, the efficiency of AI-driven models is equally critical, especially in real-time applications. Therefore chapter 3 looks into strategies to reduce latency in detection when dealing with ultra-high-resolution (UHR) images. Here, we explore this challenge through the problem of small barcode detection in UHR images. This chapter introduces a new detection pipeline that integrates a modified Region Proposal Network (RPN) with a proposed segmentation network, Y-Net. By optimizing the learned representation to reduce latency while maintaining high accuracy, this study showcases AI's potential to handle complex visual tasks in real time, which is needed for applications requiring both speed and precision.

In chapter 4, we investigate the integration of representation learning with active perception in robotics. Here, we investigate a cross-robot self-supervised sensorimotor pre-training approach using RPTx, a non-auto-regressive multimodal model not using language that leverages the Open X Embodiment (OXE) dataset. This allowed us to gain valuable insights into the robustness and adaptability of robotic systems, including methods for reliability assessment and an examination of negative transfer instances where the model struggles to adapt. Additionally, we speculate on future directions for robotic perception, noting that the use of large language models in the pipeline has led to more stable and promising outcomes. We then contrast this in chapter 5 with our findings on LLARVA, an auto-regressive multimodal model using language, on the same Open X Embodiment (OXE) dataset, and show how this approach improves learning and facilitates generalization.

Finally, in chapter 6, we explore active perception within the geospatial domain, focusing on the integration of vision and language for flexible, user-driven tasks. We introduce LISAT, a vision-language model that learns from both imagery and text to perform tasks such as answering questions, describing images, and segmenting objects based on natural language prompts. This chapter demonstrates how multimodal representation learning enables systems to respond to dynamic, context-aware goals, providing new opportunities for decision-making in complex environments. The model is trained on curated datasets, **PreGRES** and **GRES**, and highlights how effective multimodal learning can adapt to user-defined tasks in real-world scenarios.

## Chapter 2

# Precision: Lithium Metal Battery Quality Control via Transformer-CNN Segmentation

### 2.1 Introduction

Lithium metal batteries (LMBs) offer high specific energy density, as Li is a light element. Furthermore, the liquid electrolyte is not needed in LMBs, and polymer or ceramic electrolytes can be used, which are inherently safer compared to organic electrolytes. However, currently, no commercially available LMB systems exist because of Li dendrite formation, resulting from inhomogeneous Li-metal plating. This section describes current methods for imaging batteries followed by a review of the main algorithms for image analysis, semantic segmentation, and battery characterization.

It is worth mentioning that this chapter makes use of material from our previous work (13). We also briefly discuss the operation of conventional batteries in Appendix A.

### Assessing Battery Quality with Imaging

Synchrotron-based hard X-ray computed tomography (XCT) has spatial resolution suitable to resolve dendrite structures that have microscale dimensions (14; 15). Lithium has a low atomic number and, therefore, a low X-ray attenuation. For example, Li dendrites will appear to be void spaces within dense polymer electrolyte materials. XCT imaging of LMB seldom recovers chemical information and relies on differences in material thicknesses and atomic numbers to differentiate Li from solid polymer electrolytes (SPE). To worsen the detection of LMB dendrites, they are porous formations that lead to large intensity variations within their volume. When the LMB is subjected to multiple charge-discharge cycles, a phenomenon known as pitting corrosion develops, and an electrode can present a combination of pits and dendrites, with both presenting similar X-ray attenuation in XCT

data. Thus, it is challenging to differentiate dendrites from pits, but also to quantify them properly.

Previous LMB *operando* studies using XCT analyzed Li metal plating and how the interphase evolves in symmetric Li-Li cells with polymer electrolytes and in the batteries with Li-metal anode (15; 16; 17; 18; 19). These studies focused on the battery design and functioning, however, they lack methods for revealing the structure of the dendrites. Most of these previous studies applied traditional thresholding algorithms to conduct segmentation that is unfortunately not reproducible when applied to new samples and is rarely applicable to the differentiation between Li metal, pits, and other materials. Alternatively, manual segmentation could be used for a selected cross-section, but it is often unfeasible for full-stack high-resolution imaging surveillance because it is a highly time and labor-intensive process (20).

## Deep Learning for Semantic Segmentation

Semantic segmentation is the computer vision task of splitting an image into different categories (21) using a data-driven model to assign a pixel-wise classification given the input image. Different models have been proposed, among them, the Fully Convolutional Networks (FCN) (22), which proposed a paradigm shift in 2015 that runs fully connected layers and includes a way to allow for each pixel to be classified from feature maps coming from convolutional layers. This builds on a series of local convolutions of preceding layers that aim to obtain a representation of multi-scale feature maps used for the classification tasks. Around the same time, (2) introduced their convolutional neural network (CNN), a new CNN-based encoder-decoder model known as U-Net, and showed that combining higher and lower features symmetrically is beneficial in obtaining better performance. Soon after, SegNet (23) and Deeplab (24) were proposed, confirming that the encoder-decoder architecture is well suited for such a task.

A lot of these works (12; 24; 25; 26) also leverage the *atrous convolution* to show that it could help capture contextual information. In particular, Y-Net (12) used three modules to improve segmentation accuracy. In addition to the *Regular Convolution Module*, and the *Pyramid Pooling Module* which allows the model to learn global information about potential locations of the target at different scales, the *Dilated Convolution Module* took advantage of the fact that the target is often shared out in the samples, which supports learning sparse features in their structure. PSPNet (26), on the other hand, adds ResNet (27) as a backbone while multi-scale feature maps are aggregated in its encoder.

Though these architectures work well, the computer vision community has increasingly seen their design shift from pure CNN-based (see Figure 2.1(a)) design with (12; 22; 23; 24; 26; 28) to transformer-based (see Figure 2.1(b;c;d;e)) designs which started with ViT (29; 30; 31; 32; 33). Later on, hybrid models (see Figure 2.1(b;c;d)) started exploiting the best of both worlds by either using a transformer as encoder and CNN as decoder (34; 35), or CNNs as encoder and a transformer as decoder, or even using CNNs and encoder-decoder while transformers are used in the middle or in between to process the feature maps. One

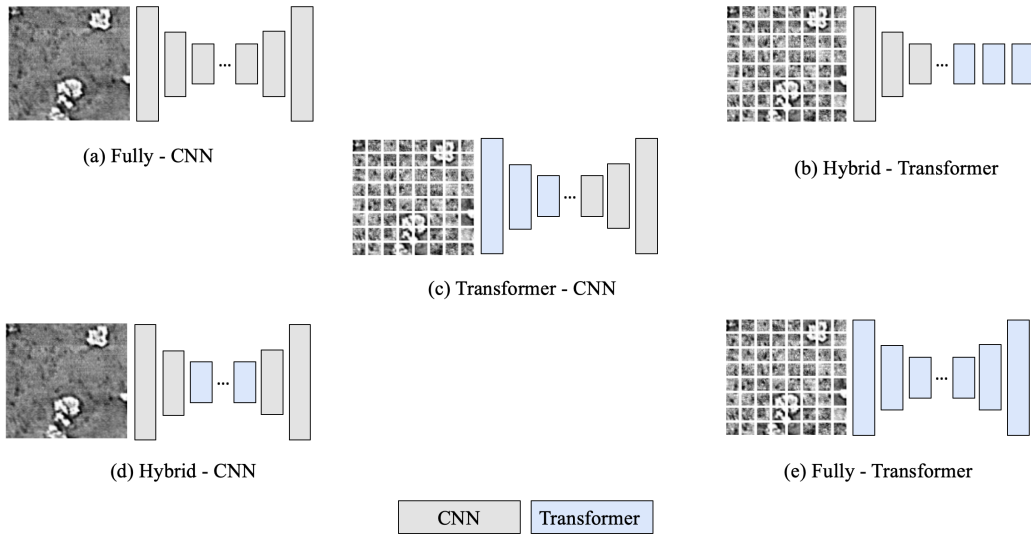


Figure 2.1: Types of Hybrid architectures.

such hybrid model is HRNet-OCR (36), which has a CNN as a backbone and combines it with cross-attention layers between features of different scales to account for multiple contexts and scales in the data. Table 2.1 gives a comprehensive comparison of the two types of model, and in this chapter, we stick to the Transformer-CNN hybrid model because of the high performance and fewer parameter counts that it offers.

Criterion	Transformer Encoders	CNN Encoders
Global Context Awareness	Capture long-range dependencies and global context; can be ideal for images with complex structures or widely spread patterns.	Focus on local features, often limited in the ability to capture global patterns.
Multi-Head Attention	Allows simultaneous processing of different parts of the image, considering various aspects of the data.	Unable to process multiple parts of the image concurrently.
Attention Mechanisms	Focus on relevant parts of the data; can handle complex and cluttered backgrounds effectively.	Use fixed receptive fields and weight sharing.

Table 2.1: Comparison of Transformer encoder vs. CNN encoder

In all these schemes, the common denominator remains the attention mechanism, which has proven in the past few years to exceed the performance of models disregarding it. That is because it allows the features not to be subject to the inductive biases and translation invariance that occur in CNNs. Instead, it allows the model to learn long-term dependencies between pixel locations (29). In other words, it allows for a better representation by leveraging contextual information either between pixels, patches, or channels.

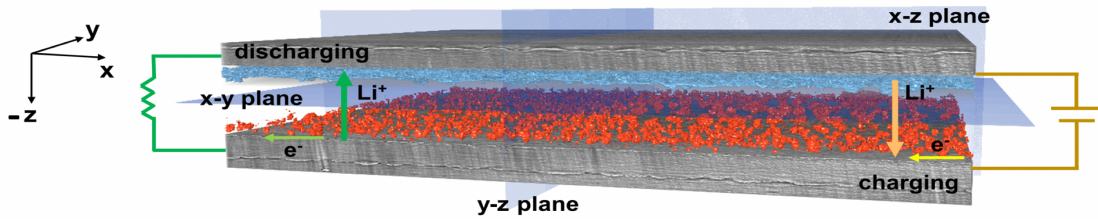


Figure 2.2: Diagram illustrating the Li-polymer-Li symmetric cell design, imaged using X-ray CT, with highlighted dendrite formations (blue) and the redeposited Li (red).

## Problem and Motivation

Li dendrite formations initiate during battery cycling as illustrated in Figure 2.2, with dendrites nucleating on the interface between the electrolyte and the electrodes. Dendrite growth depends on the current density of plating, electrolyte transference number, electrolyte mechanical properties, and impurities present in Li-metal material and at the interfaces. Earlier works have shown that increasing the shear modulus of the SPEs can help suppress Li dendrite growth but cannot fully eliminate it (37; 38; 39; 40). Further studies have shown that Li-metal surface impurities ( $\text{Li}_3\text{N}$ ,  $\text{Li}_2\text{CO}_3$ , and  $\text{Li}_2\text{O}$ ) can result in inhomogeneous current density and promote nucleation of Li dendrites.

Accurate segmentation for measuring dendrite volume has guided research and quality control of battery designs, as well as tests of materials used for its components. Deep learning methods can provide exceptional segmentation results (41; 42; 43) when using high-resolution XCT data, particularly when large collections of annotated data are available. For example, (44) used a CNN known as D-LinkNet to inspect the effect of distortion on the segmentation accuracy of Li-ion batteries. Additional works by the same team (45) used a U-Net for multiphase segmentation of battery electrodes from nano-CT images. Despite being focused on battery segmentation, those studies lack information on dendrite segmentation.

Previous studies (46; 47) on inspecting dendrites in batteries discussed problems regarding the mechanisms and types of nucleation, e.g., lateral growth or Li filaments. Data acquisition modes range from electron microscopy (46; 48; 49) to XCT (47; 49) with valuable morphological characterization and designs for suppression of dendrite growth, but dendrite detection was addressed mostly qualitatively through dendrite projections and/or visualizations. For example, the dendrite volume calculation in (49) was based on median filtering and Otsu thresholding, a method that seldom works for more than a few slices from an XCT stack, unless considering strenuous manual post-processing (50).

## Research Contributions

The proposed method describes the design and implementation of precise dendrite and Li deposit segmentation from 3D XCT images. In performing the segmentation of a 3D volume automatically, we could either design a 3D model that performs directly on an input volume, or we could leverage 2D models by subdividing the volume into slices. This chapter introduces a 2D model due to its ability to be trained faster and with a limited number of samples, hence making it more versatile. In particular, we propose an architecture that benefits from both the contextual information learned from transformers and the global information captured by CNNs to predict dendrites and Li deposits from a lithium metal battery that underwent cycling and was imaged using high-resolution XCT data. We compare four different deep learning architectures, including U-Net, Y-Net, TransforCNN (T-Net), and E-Net, on their ability to segment dendrites inside the cycled symmetric Li-Li battery with polymer electrolytes.

## 2.2 Materials Description

Li metal batteries have several benefits over Li-ion batteries as shown in Table 2.2. It holds a high theoretical capacity (3860 mAh/g) and a large negative thermodynamic potential (-3.06 V vs. SHE)(51). Thus, it is considered a promising candidate for the next-generation battery anode. Li metal is highly active and can introduce a series of side reactions in a battery system with liquid electrolytes. This can also cause the dendrite to form, which would eventually lead to short circuits and bring safety issues to the battery. Using solid electrolytes, instead of liquid electrolytes, a more stable interface can be designed between the electrolyte and the Li metal electrodes, thus alleviating the dendrite formation issue. Recent developments in electrolyte engineering can be found in (45; 49; 52; 53).

Currently, several solid-state materials are used as electrolytes and separators, such as polymers and single-ion conducting inorganic solid electrolytes (glass or ceramic). Polymer materials are promising as they are mechanically flexible, because polymer materials can be produced in a roll-to-roll scalable process and be designed very thinly. However, for SPEs to have broad deployment, strategies for dendrite suppression must be developed, such as coatings and soft interlayers including polymers as well as ionic liquids (54; 55; 56).

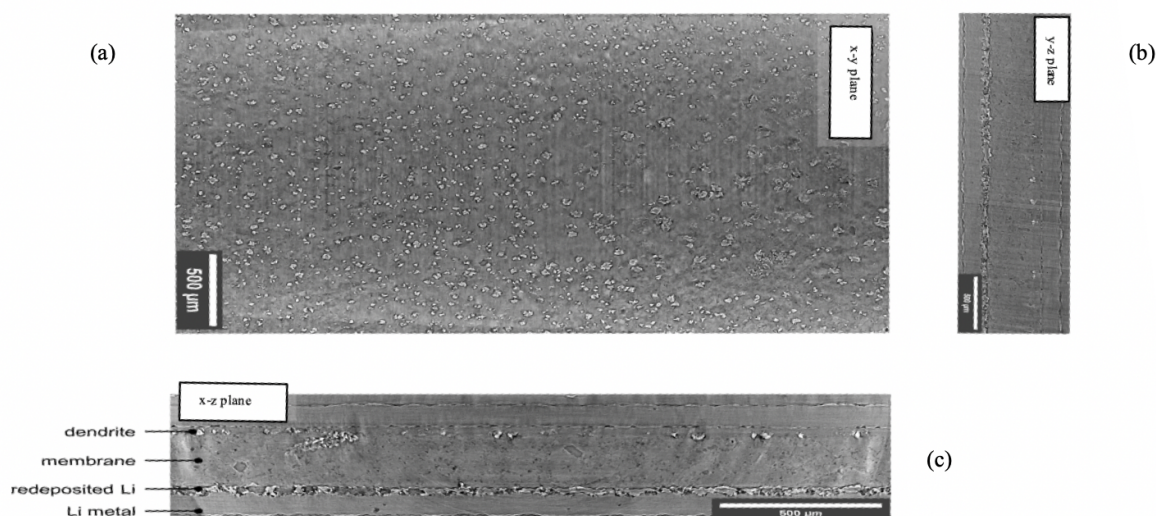


Figure 2.3: Cross-Sectional Images for the Li-polymer-Li symmetric cell; (A) Cross section of the x-y plane where the training was done on this plane; (B) Cross-sections of the x-z plane and detailing of the cell components; (C) Cross-sections of the y-z plane.

Aspect	Lithium-ion Batteries	Lithium Metal Batteries
Status	Mass Production	Active Development/Improvement
Energy Density	High energy density, suitable for various applications	Higher energy density, potentially more space-efficient
Safety	Prone to dendrite formation, Heat, Liquid Electrolyte	Prone to dendrite formation
Electrolyte type	Liquid, more risks	Solid, Safer
Cost	More expensive to manufacture	Potentially lower manufacturing cost depending on scale
Charging Speed	Moderate charging speed	Potentially faster charging due to higher conductivity

Table 2.2: Comparison of Li-ion and Li-metal Battery.

Design of interfaces in Li-metal batteries or All Solid-State Batteries (ASSBs) is challenging, as it involves control of Li-ion plating onto Li metal, and this plating process needs



to be uniform. When not done properly, interactions at the interfaces of electrodes arise due to transport processes associated with ions (53; 57), which can form ionic aggregates. Dendrite growth can be triggered by the formation of localized regions of high lithium ion concentration, which can occur due to the clustering of lithium ions into ionic aggregates. In general, dendrites are formed during battery charge and discharge cycles. This happens especially when a battery is charged at high current densities, due to the heterogeneous Li metal plating, even when considering solid electrolytes. Tracking the structure and evolution of dendrites is important to develop a strategy to prevent their growth. Dendrites are tree-like and porous structures, usually with a size in nano to micro-scale. Given the morphological structure of dendrites and their size relative to an input stack, performing XCT segmentation is a suitable method of analysis as it allows for a pixel-wise classification, which helps quantify the volume of dendrites and use it as a proxy of battery quality.

## Electrochemical Testing

Li/Li symmetric cells are assembled using two Li metal electrodes and are considered a tool for testing and observing the Li metal anode without being affected by cathode materials.

Free-standing Li-metal foils with a thickness of 100  $\mu\text{m}$  from FMC were used. Polymer electrolyte membrane was sourced from an industrial partner with a thickness of 140  $\mu\text{m}$  as a research sample. The cell was assembled as Figure 2.4 shows. A red shim with a thickness of 50  $\mu\text{m}$  was used to create a circle with a 0.8 cm diameter. Two circular polymer electrolytes with a diameter of 0.80 cm were punched out and placed on each side of the red shim. Two Li-metal foils were then placed on the outside of the membranes as electrodes. The electrodes were connected to the metal tabs. The cell was sealed with a vacuum sealer. A current density of 1.5  $\text{mA}/\text{cm}^2$  was periodically applied to the cell for 10 minutes, and the battery rested for 20 minutes. The cell was cycled at 3.0  $\text{mAh}/\text{cm}^2$  for one full cycle (120 minutes for charging and 120 minutes for discharging), after which the cell XCT scan was acquired.

## Synchrotron X-ray CT Imaging

The XCT scan was acquired at Beamline 2-BM at Advanced Photon Source (APS) at Argonne National Laboratory (ANL), which used a 20  $\mu\text{m}$  LuAG scintillator with  $5\times$  lenses, and an sCMOS PCO edge camera. A 27.5 keV energy was selected using a multilayer monochromator, with 100 ms exposure time per back-projection and over 180 degrees of rotation, enabling the collection of 1,500 projections as depicted in Figure 2.5 from (1). The yielded image presents a resolution of 1.33  $\mu\text{m}/\text{pixel}$  and a field-of-view of 3.3 mm. Three FOVs were recorded and were stitched together to form a vertical height of  $>3$  mm during the post-processing. Tomographic reconstructions considered TomoPy with the Gridrec algorithm(58; 59; 60).

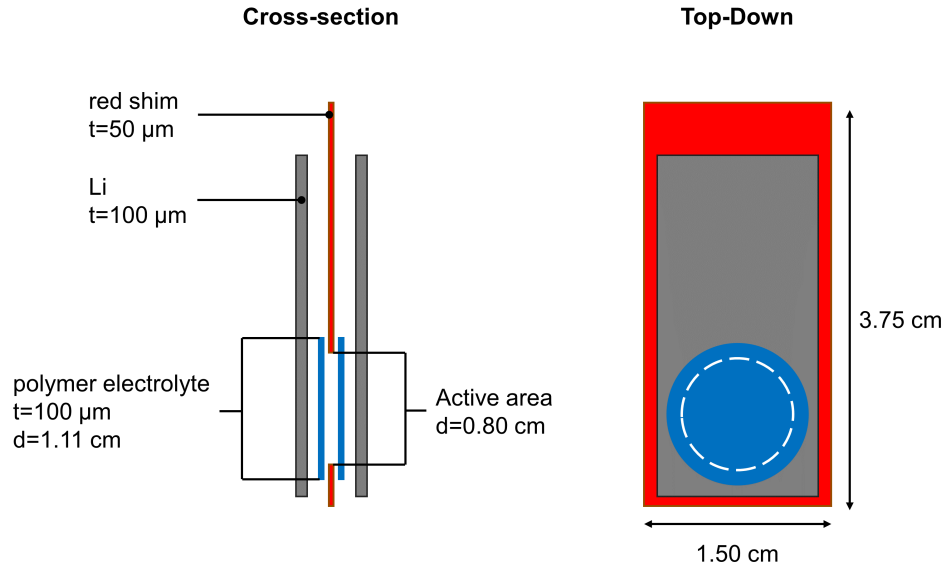


Figure 2.4: Schematic illustration of the pouch cell.

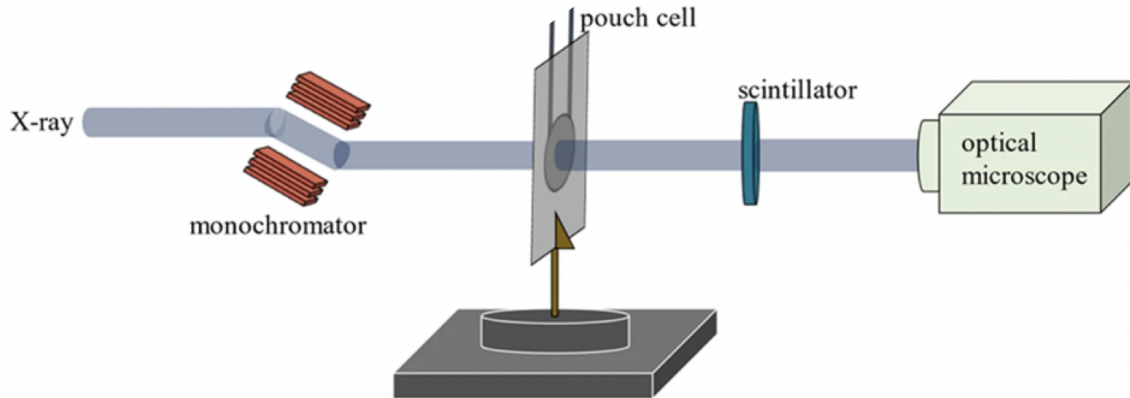


Figure 2.5: Schematic illustration of the dataset collection process. This process was outlined in (1)

## Raw data Pre-Processing

The resulting raw TomoPy reconstruction was a large volume of size  $(3977, 2575, 2582)$  and was not properly aligned as expected due to the various motions involved in collecting the

data, as shown in Figure 2.6. As this raw data contains a lot of noise and irrelevant parts, we proceed to develop an algorithm that will allow us to cleanly crop out those regions. In the process, we inverted the grayscale volume and obtained a maximum projection image on the stacks to facilitate our ability to locate corners.

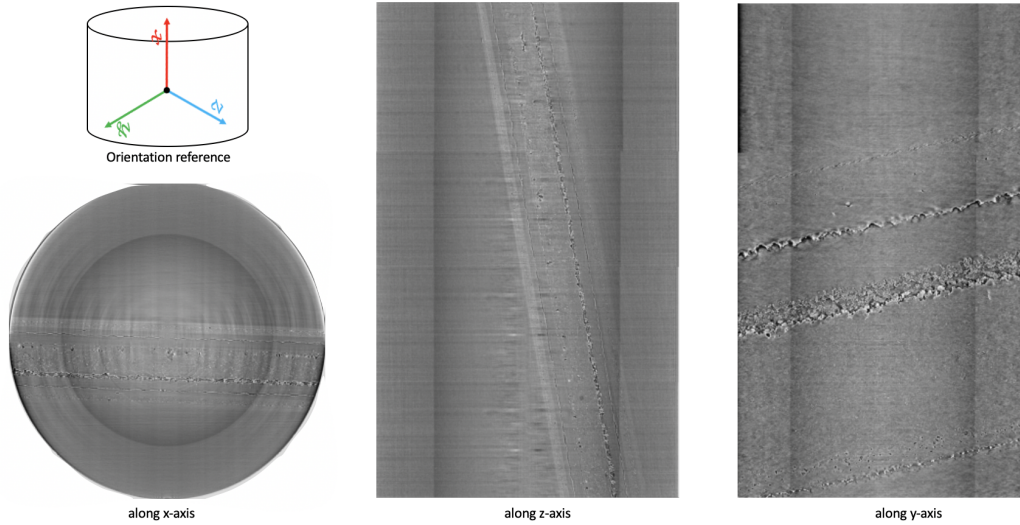


Figure 2.6: Sample Raw data obtained after TomoPy reconstruction of a CT scan

To align the data, we first used a series of perspective transformations and homography to rectify the region of interest along each plane. Though this process could be automated using feature detectors and feature matching techniques such as MOPS (61) and SIFT (62), we manually selected corners for optimal precision. We then rectified and cropped the raw data to obtain the region of interest to a volume of size  $(3849, 340, 2071)$  shown in Figure 2.7.

## 2.3 Computational Methods

For a given volume stack, we apply 2D models due to their versatility. In doing so, we subdivide each training hand-labeled slice into  $128 \times 128$  patches which were then separated into training, validation, and testing datasets. Based on their known performances over the years, we investigate CNN encoder-based architectures such as U-Net (28) and Y-Net (12). In addition, we also compared their performance with TransforCNN, our proposed Transformer encoder-based network, and E-Net, an Ensemble Network over U-Net, Y-Net, and TransforCNN. We train the networks in a weakly-supervised fashion where a small subset of labeled data is used in conjunction with a much larger unlabeled sample size. Figure 2.12 shows a sample output of the model considered in this chapter.



Figure 2.7: Sample Region of Interest (RoI) data obtained after pre-processing TomoPy reconstruction of a CT scan

## U-Net

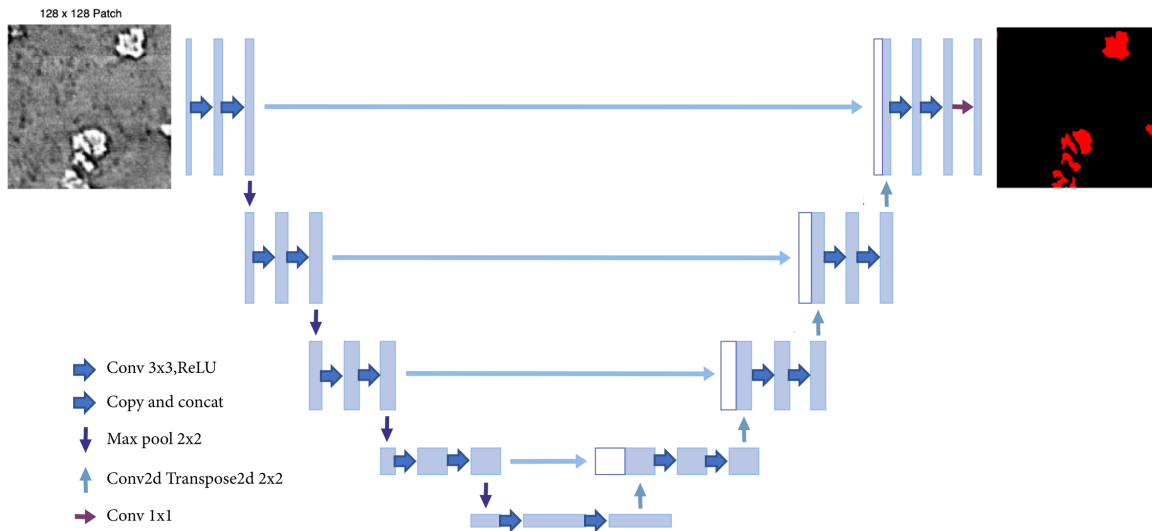


Figure 2.8: U-Net architecture adapted from (2).

U-Net is a CNN architecture that was first introduced in 2015 by (2) for the semantic

segmentation of biomedical images. We refer interested readers to (2) for details about the architecture of the network. In this chapter, the model was adapted to take images of size  $128 \times 128$  as input. For the encoder, we started with 16-channel  $3 \times 3$  kernels and doubled the number at each layer, followed by a ReLU activation and max pooling until a 256-channel  $8 \times 8$  resolution feature map is obtained. We reversed the operation with transposed convolutions operation on the decoder side and concatenated with corresponding-sized encoder feature maps until the desired output shape was obtained.

## Y-Net

Y-Net is a CNN architecture, detailed in Chapter 2, which was originally introduced by (12) to tackle latency issues in barcode segmentation from Ultra High-Resolution images. By leveraging Y-Net’s architecture shown in Figure 3.6, we have modified and adapted the *Regular Convolution Module* to take in  $128 \times 128$  images from training slices. As it consists of convolutional and pooling layers, we started with 24-channel  $3 \times 3$  kernels and doubled the number at each layer. We alternated between convolution and max-pooling until we reached a feature map size of  $8 \times 8$  pixels. The *Dilated Convolution Module* here took advantage of the fact that dendrites are often shared out in the samples to learn sparse features in their structure. It also took  $128 \times 128$  input patches, and we maintained 16-channel  $3 \times 3$  kernels throughout the module while the dimensions of the layers were gradually reduced using a stride of 2 until a feature map of  $8 \times 8$  pixels was obtained. Finally, the *Pyramid Pooling Module*, which allows the model to learn global information about potential locations of the dendrites at different scales, had its layers concatenated with the layers on the dilated convolution module to preserve the features extracted from both modules.

## TransforCNN

TransforCNN is a hybrid Transformer-CNN segmentation model that leverages the encoder model of Vision Transformers ViT (29) and the decoder architecture of CNNs. More specifically, its encoder model was first introduced in Natural Language Processing (NLP) by (63), and its multi-headed self-attention was later shown (by ViT) to help remove the common inductive biases observed in CNN-only models by relating all input sequences with each other. As depicted in Figure 2.9, the proposed architecture is hybrid because it combines the Transformer Encoder Block with the CNN Decoder Block to deliver semantic segmentation.

The ***Transformer Encoders Block*** takes inputs that are  $16 \times 16$  sub-patches sequences from the  $128 \times 128$  patches that were obtained from training slices. These patches are flattened and each is embedded into a 64-dimensional feature vector via a linear projection and is added to its corresponding Fourier Features (FF) positional encoding. We used 8 transformer encoder units, and the outputs of every 2 transformer encoders were reshaped into a 2-dimensional feature map representation, concatenated, up-sampled recombined with

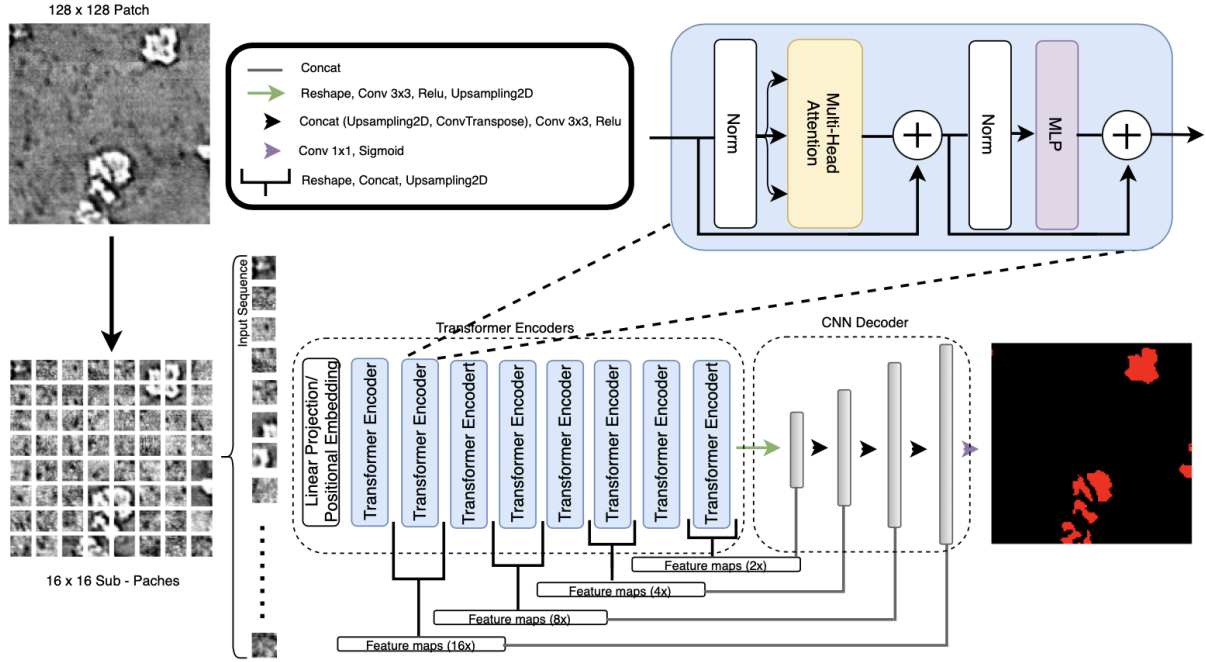


Figure 2.9: T-Net architecture.

the layers from the CNN Decoder Block of corresponding dimension.

The **CNN Decoder Block** takes in the output of the last transformer encoder unit, reshapes it into a 2-dimension representation on which a set of  $3 \times 3$  kernels convolutions and max-pooling is applied to obtain a feature map of  $8 \times 8$  pixels. The resulting feature maps are then concatenated with corresponding size feature maps coming from the Transformer Encoder Block and up-sampled continuously until the final output is obtained. This last step allows for the enhancement of the features in the CNN Decoder Block as we are progressively reconstructing the output dimension of  $128 \times 128$ .

## E-Net

We have combined the results of different architectures, namely U-Net, Y-Net, and TransforCNN, to create an ensemble prediction scheme called E-Net. It was found that our best mean Intersection over Union (mIoU) is obtained when combining 20% of U-Net with 80% of TransforCNN while the best mean Dice Similarity Coefficient (mDSC) is obtained using only a TransforCNN.

## 2.4 Experimental Results

In training the models (U-Net, Y-Net, and TransforCNN), we used one NVIDIA Tesla V100 GPU for each experiment. We obtained a total of 4433 samples of resolution  $128 \times 128$  with their corresponding hand-labeled ground truth that the models were trained on. We used 80% of the examples for the training set, 10% for the validation set, and 10% for the testing set. We used data augmentation schemes in training all models which consist of random rotations in all directions, random flips (vertically and horizontally), random cropping (2%), random shifts, random zoom (range in  $[0.8, 1]$ ), and a small range of random brightness and contrast variation  $x \pm 5\%$ . We trained the U-Net for 450 epochs while the Y-Net and TransforCNN models were trained for 130 and 300 epochs, respectively.

As shown in Figure 2.10, the Y-Net and TransforCNN converge faster than U-Net with the initial loss of the TransforCNN model being significantly lower than that of the U-Net and Y-Net models. We have experimented with various loss functions such as Tversky loss (64) described in Eq. 2.1, the focal Tversky loss (65) described in Eq. 2.2, the balanced cross-entropy loss described in Eq. 2.4, and the binary cross-entropy loss (Eq. 2.3) out of which the latter yields the best results. One interesting observation is that though the validation curve on U-Net exhibits characteristics of a better generalization, the quantitative results show otherwise.

For evaluation, we have used the dice similarity coefficient described in Eq. 2.5 and the Jaccard Index also known as Intersection over Union Eq. 2.6. In all the equations,  $y$  and  $\hat{y}$  are respectively the ground truth and prediction on patch  $i$ , and TP, FP, and FN represent the number of true positives, false positives, and false negatives, respectively.

$$\mathcal{L}_{\text{Tversky}}(y, \hat{y}) = \frac{y\hat{y}}{y\hat{y} + \beta(1-y)\hat{y} + (1-\beta)y(1-\hat{y})} \quad (2.1)$$

where  $\alpha, \beta > 0, \alpha + \beta = 1$ .

$$\mathcal{L}_{\text{Focal Tversky}}(y, \hat{y}) = [1 - \mathcal{L}_{\text{Tversky}}(y, \hat{y})]^\gamma \quad (2.2)$$

where  $\alpha, \beta > 0, \alpha + \beta = 1, \gamma = 4/3$ .

$$\mathcal{L}_{\text{Cross Entropy}}(y, \hat{y}) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y}) \quad (2.3)$$

$$\mathcal{L}_{\text{balanced Cross Entropy}}(y, \hat{y}) = -\beta y \log(\hat{y}) - (1-\beta)(1-y) \log(1-\hat{y}) \quad (2.4)$$

where  $\beta \in [0, 1]$ .

$$\mathcal{DSC}(y, \hat{y}) = \frac{2TP}{2TP + FP + FN} \quad (2.5)$$

$$\mathcal{IoU}(y, \hat{y}) = \frac{TP}{TP + FP + FN} \quad (2.6)$$

$$(2.7)$$

Note that Eq. 2.6 could also be expressed as  $\mathcal{DSC} / (2 - \mathcal{DSC})$ .

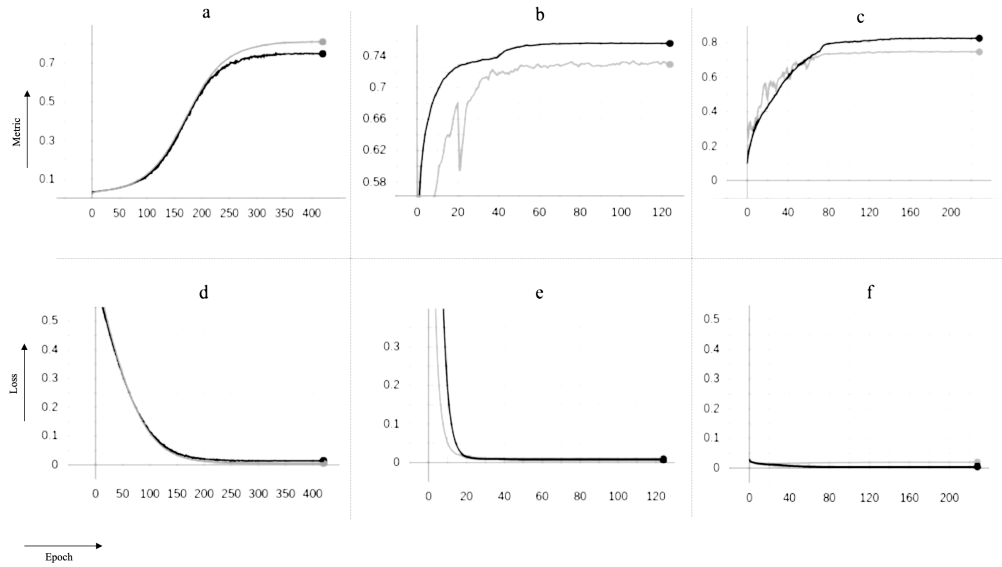


Figure 2.10: Training curves for U-Net, Y-Net, and TransforCNN on the y-axis vs numbers of epochs on the x-axis; the models were optimized over the binary cross-entropy function as loss and evaluated on the dice similarity coefficient as evaluation metric during training; The gray curves depict behavior on the validation sets while the black curves show behavior on the training sets over increasing numbers of epochs; (a) training and validation dice coefficient for U-Net; (b) training and validation dice coefficient for Y-Net; (c) training and validation Dice coefficient for TransforCNN; (d) training and validation loss for U-Net; (e) training and validation loss for Y-Net; (f) training and validation loss for U-Net; the use of dropout during only the training phase explains why the models tend to perform better on the validation set over time.

For all of the models, we use the mean Dice Similarity Coefficient (mDSC) and the mean Intersection over Union (mIoU) as metrics, shown in Table 3.1. As seen, our proposed pipeline outperforms U-Net (28), and Y-Net (12) by a  $mIoU$  of 8.13% and 10.3% and  $mDSC$



	mIoU	mDSC	latency (ms)	Input Patch Resolution (px)
U-Net	.8698	.8998	<b>65.36</b>	$128 \times 128$
Y-Net	.8481	.8790	103.62	$128 \times 128$
T-Net	.9511	.9647	206.75	$128 \times 128$
E-Net	.9514	.9641	473.59	$128 \times 128$

Table 2.3: mIoU, mDSC, Inference Time, and Patch Size for Li-Li Symmetric battery Dataset.

	Volume (Cubic Px)	Occupied Volume Percentage
U-Net	54,940,997.0	2.027
Y-Net	99,389,447.0	3.667
TransforCNN	82,892,014.0	3.058
E-Net	80,858,216.0	2.983

Table 2.4: Volume and Percentage Volume Occupied for U-Net, Y-Net, TransforCNN, and E-Net.

of 6.49% and 8.57%, respectively. Also shown in Table 3.1 is a slight *mIoU* improvement of 0.03% by our Ensemble Network analysis (E-Net) on TransforCNN.

In addition, Table 3.1 displays that while TransforCNN is successful in segmenting out dendrites, its latency is at least  $3.16\times$  slower than U-Net, which has the fastest latency of all models evaluated at 65.36 milliseconds (ms). The slowest of all the models is observed to be E-Net, which performs  $7.24\times$  slower than U-Net.

Qualitatively, Figure 2.11 shows the predictions on a given test slice and more specifically, Figure 2.12 shows sample predictions at the patch level. As observed in the first and second rows (a;b), the TransforCNN and U-Net predictions are the closest to the Ground Truth. The third, fourth, and eighth rows (c; d; h) show that U-Net and Y-Net tend to generalize better as the unlabeled dendrite regions in the input patches are segmented out by these two models, while TransforCNN and E-Net still reflect the Ground Truth images. The fifth, sixth, and seventh rows (e; f; g) show the generalization potential of all models, while the predictions of TransforCNN and E-Net overall tend to remain closer to the Ground Truth. In addition, Table 3.2 summarizes the absolute number of voxels corresponding to the segmented dendrite and re-deposited Li volume as well as their volume fraction.

## 2.5 Discussion

LMBs are promising candidates for next-generation batteries because of their high specific energy density. Currently, Li dendrites growth is an issue, as it can lead to loss of Li (dead Li), shorting of cells, and other undesirable degradation phenomena.

More specifically, short-circuiting is a leading failure mechanism in LMBs due to the uncontrolled propagation of lithium protrusions that often present a dendritic morphology. The energy density benefits of using LMBs can only be harvested after scientists are able to detect and regulate the dendrite formation and control dendrite growth.

Uneven lithium-ion distribution and dendrite formation can compromise battery performance and safety. For this reason, this chapter introduced a semantic segmentation algorithm called TransforCNN that detects both dendrites and re-deposited Li accurately and compares it with traditional approaches.

Overall, it was observed that TransforCNN and E-Net tend to learn semantics in the Ground Truth images provided during training. In contrast, U-Net and Y-Net tend to simply generalize even to cases where segments in the Ground Truth were wrongly hand-labeled. We speculate that this may lead to U-Net and Y-Net being wrongly penalized during the evaluation process, while TransforCNN and E-Net are rewarded since their predictions always look the closest to the Ground Truth.

Experiments have also illustrated that our approach outperforms existing methods, though it is slower than the fastest (U-Net) of all considered models. In future work, extending this method to a multi-class segmentation task, differentiating dendrites from pits and bubbles while improving the current latency in a weakly supervised fashion.

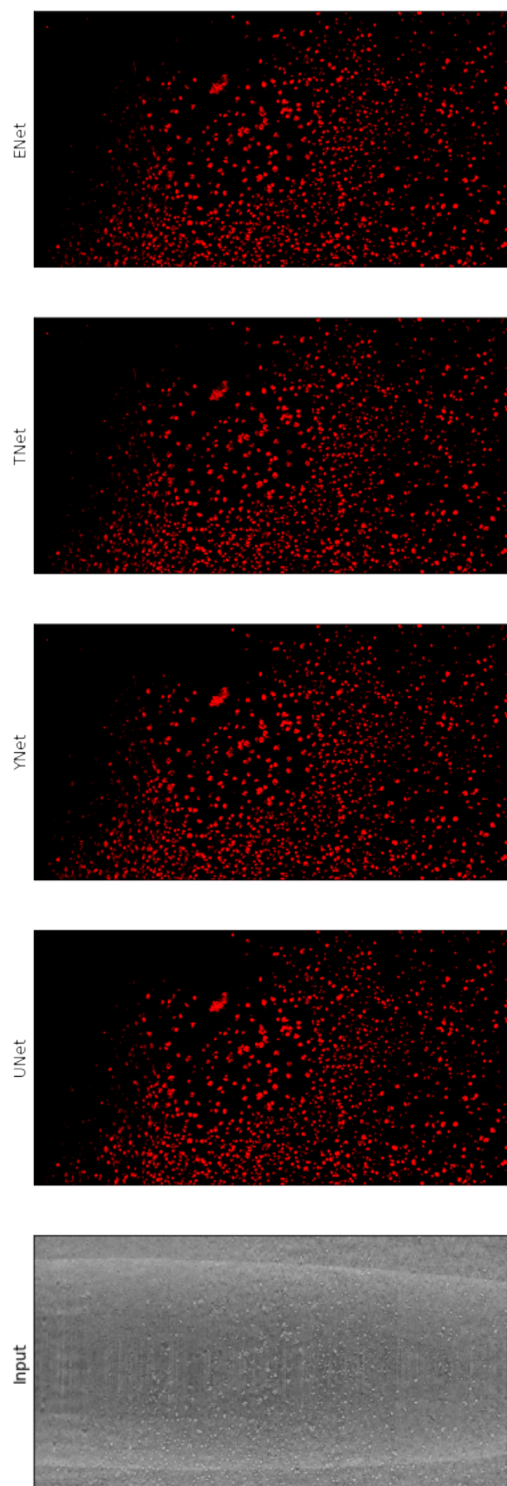


Figure 2.11: U-Net, Y-Net, T-Net, and E-Net sample outputs on a slice showing a cross-section along the x-y plane.

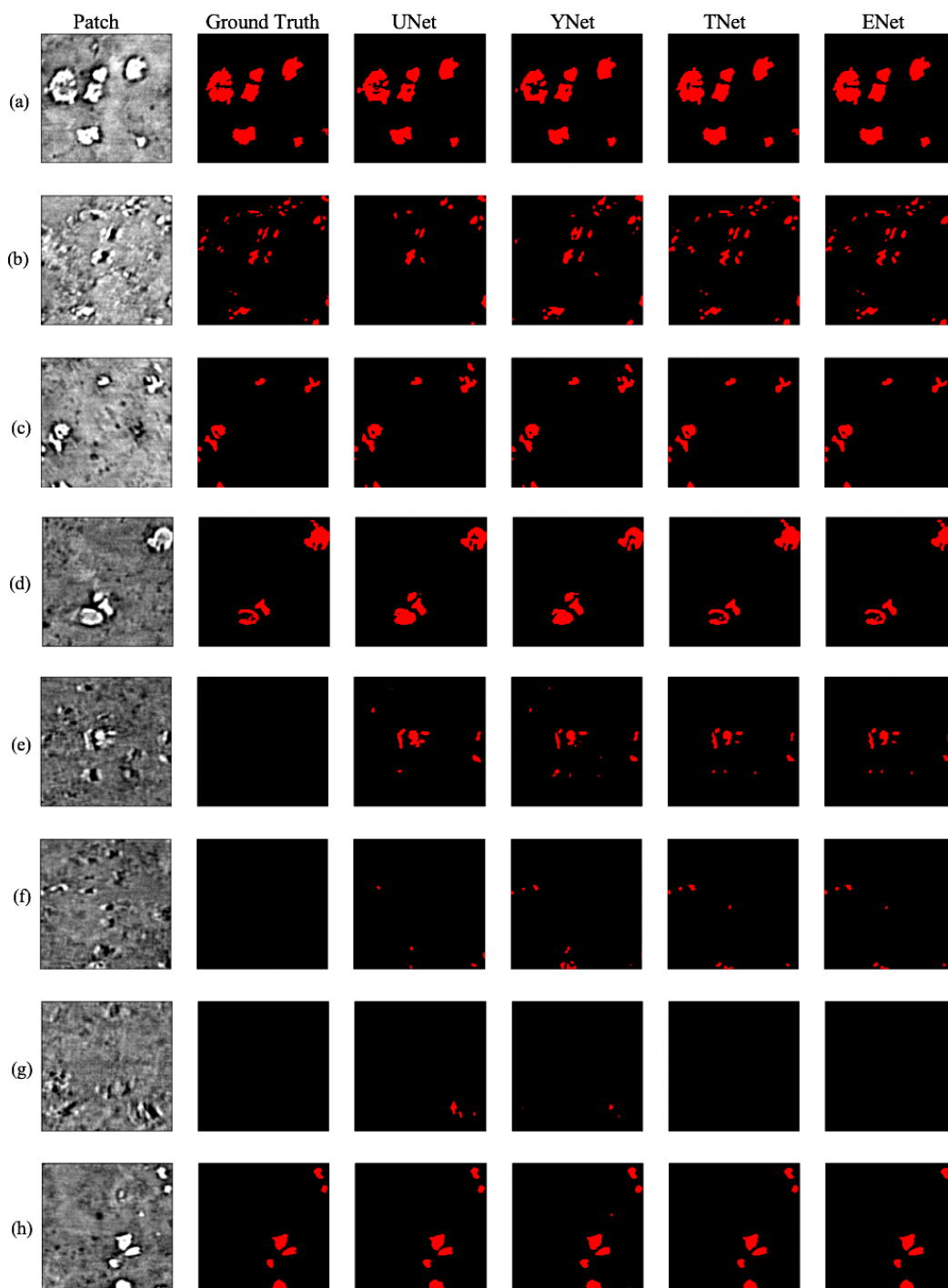


Figure 2.12: U-Net, Y-Net, T-Net, and E-Net sample outputs on random test patches.

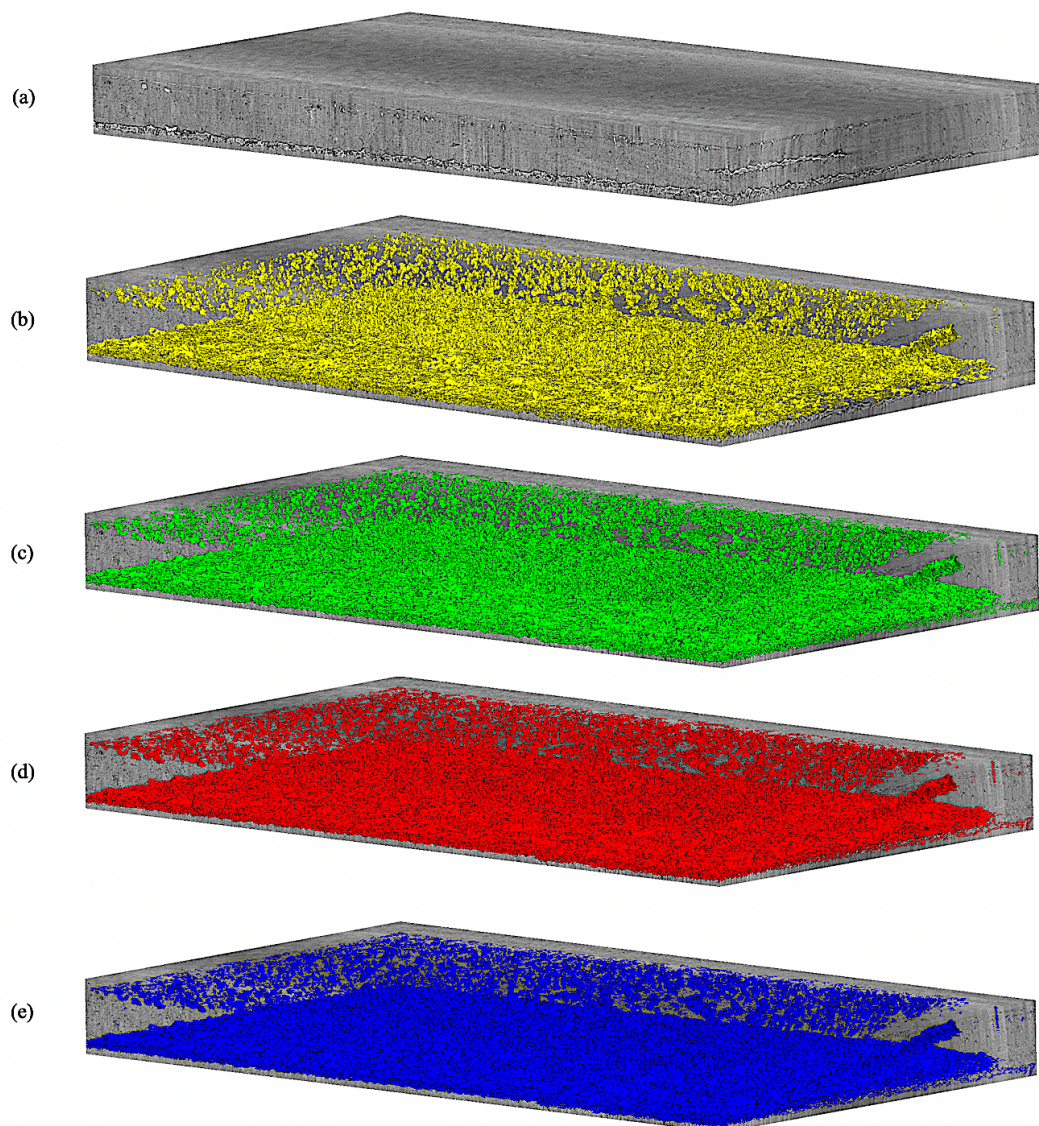


Figure 2.13: 3D rendering of U-Net, Y-Net, T-Net, and E-Net on test volume; (a) grayscale test input volume; (b), (c), (d), (e) are respectively U-Net, Y-Net, T-Net, and E-Net predictions



## Chapter 3

# Latency: Fast, Accurate Barcode Detection in Ultra-High-Resolution Images

### 3.1 Introduction

Barcodes are digital signs often made of adjacent and alternating black and white smaller rectangles that have become an intrinsic part of human society. In administration, for example, they are used to encode, save, and retrieve various users' information. At grocery stores, they are used to track sales and inventories. More interestingly, in e-commerce, they are used to track and speed up processing time in warehouses and fulfillment centers.

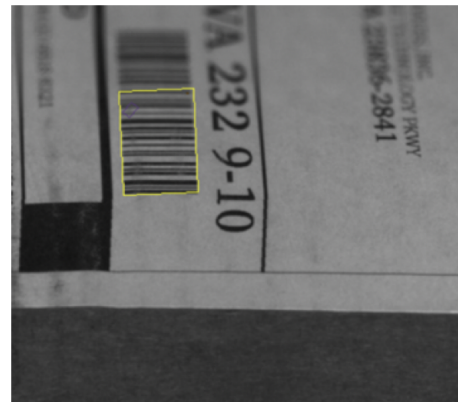


Figure 3.1: Barcode detection with classical Signal Processing.

In classical signal processing, filters used for detection are image-specific since input images are not all necessarily acquired with the same illumination, brightness, angle, or

camera. This causes such algorithms to have a high failure rate, as shown in Figure 3.1. Consequently, adaptive image processing algorithms are required, which can impact detection accuracy (66). In addition, because classical signal processing methods often run on Central Processing Units, they tend to be much slower compared with deep learning implementations that are easily optimized on Graphics Processing Units (GPUs).

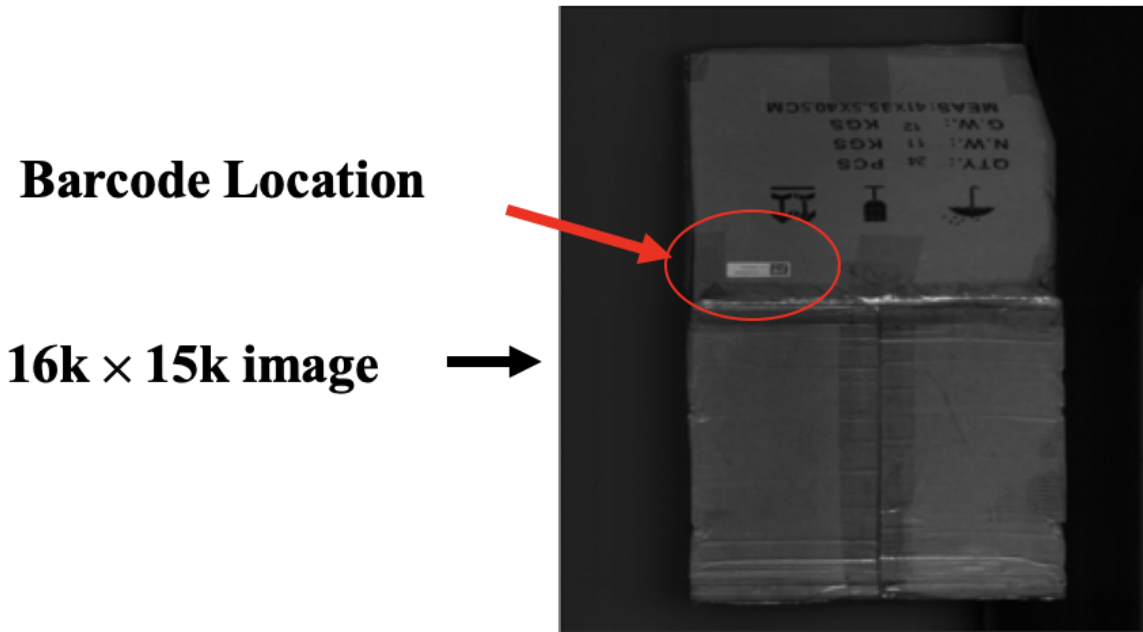


Figure 3.2: Barcode detection in UHR images.

Over the years, a number of methods have been proposed to detect barcodes using classical signal processing (66; 67; 68; 69; 70), but nearly all of them take too long to process Ultra High-Resolution (UHR) images because the image is too large while the actual barcode location may occupy a small region in the image (see Figure 3.2). More specifically, (70) used parallel segment detectors which improved on their previous work (71) of finding imaginary perpendicular lines in Hough space with maximal stable extremal regions to detect barcodes. (68) used morphological manipulation for barcode detection, but this method did not generalize well as different barcode types have varying detection performances. Similarly, (72) proposed using  $x$  and  $y$  derivative differences, but varying input images yield different outputs, and using such an operation on UHR images often becomes highly inefficient.

With neural networks, though there has been much improvement in barcode detection tasks, few of them have addressed the fast and accurate detection problem in UHR images.

(73) paved the way for using neural networks to detect barcodes by investigating Hough spaces. This was followed by (74), which adapted the You Only Looked Once (YOLO) detector to find barcodes in Low Resolution (LR) images, but the YOLO algorithm is known to perform poorly with long shaped objects such as Code 39 barcodes. Instance segmentation methods such as Mask R-CNN (75) perform better on images of size  $1024 \times 1024$  pixels, but on smaller size images, the outputted Region of Interests (RoI) do not align well with long, 1D-barcode structures. This is because it typically predicts masks on  $28 \times 28$  pixels irrespective of object size, and thereby generates "wiggly" artifacts on some barcode predictions, losing spatial resolution. In the same way, dedicated object detection pipelines, such as YOLOv4 (76), though they perform well on lower Intersection over Union (IoU) thresholds, suffer accuracy at higher IoU thresholds. Among those using segmentation on LR images as a means for detection, (77) also tends not to perform well at higher IoU thresholds.

In this chapter, incorporating material from our work in (78), we propose a pipeline for detecting barcodes using deep neural networks, shown in Figure. 3.3, which consists of two stages trained separately. When compared with classical signal processing methods, neural networks not only provide a faster inference time but also yield higher accuracy because they learn meaningful filters for optimal feature extraction. As seen in Figure. 3.3, in the first stage, we expand on the Region Proposal Network (RPN) introduced in Faster R-CNN (79) to extract high-definition regions of potential locations where barcodes might be. This stage allows us to significantly reduce inference computation time that would have been required otherwise in the second stage. In the second stage, we introduce Y-Net, a semantic segmentation network that detects all instances of barcodes in a given outputted RoI image ( $400 \times 400$ ). We then apply morphological operations on the predicted masks to separate and extract the corresponding bounding boxes as shown in Figure. 3.5.

One of the limitations of existing work on barcode detection is the insufficient number of training examples. ArTe-Lab 1D Medium Barcode Dataset (73) and the WWU Muenster Barcode Database (80) are two examples of existing available datasets. They contain 365 and 595 images, respectively, with ground truth masks at a resolution of  $640 \times 480$ . Most of the samples in the ArTe-Lab dataset have only one EAN13 barcode per sample image, and few of them in the Muenster database have more than one barcode instance on a given image. To address this dataset availability problem, we have released 100,000 UHR and 100,000 LR synthetic barcode datasets along with their corresponding bounding boxes ground truths and their ground truth masks to facilitate further studies. The outline of this chapter is as follows: in Section 2, we describe details of our approach; in Section 3, we summarize our experimental results; and in Section 4, we conclude and expand on our future work.



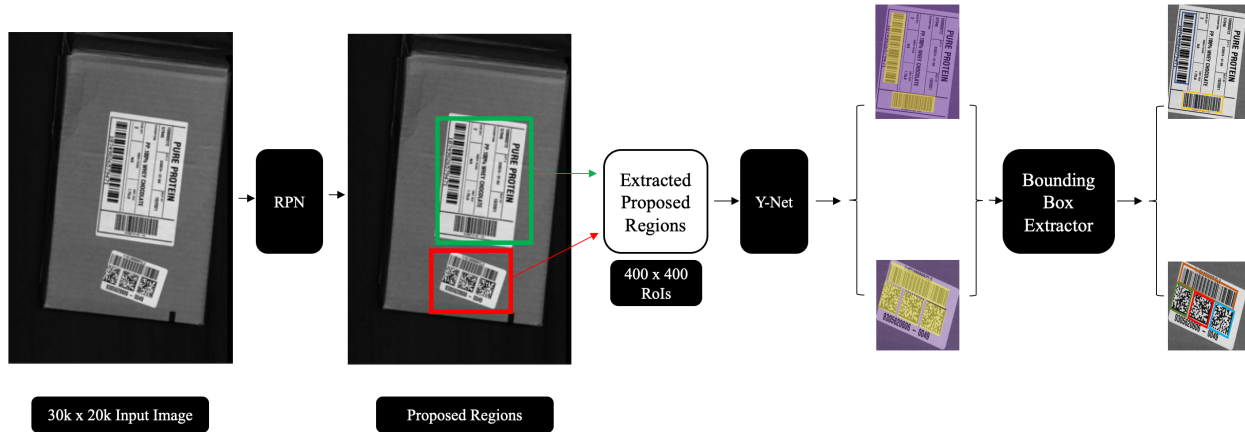


Figure 3.3: Proposed approach, the modified RPN is followed by Y-Net and the bounding box extractor.

## 3.2 Proposed Approach

As seen in Figure. 3.3, our proposed method consists of three stages: the modified Region Proposal Network stage, our Y-Net<sup>1</sup> segmentation network stage, and the bounding box extraction stage.

### Modified Region Proposal Network

Region proposals have been influential in computer vision, and more so when it comes to object detection in UHR images. It is common in UHR images that barcodes are clustered in a small region of the image. To filter out most of the non-barcode backgrounds, we modified the RPN introduced in Faster R-CNN (79) to propose regions of barcodes for our next stages. By first transforming the UHR input image to an LR input image of size  $256 \times 256$ , the RPN was trained to identify blobs in LR images. Once a bounding box is placed around the identified blobs, the resulting proposed bounding box is remapped to the input UHR image by a perspective transformation, and the resulting regions are cropped out. The LR input to the RPN is chosen to be of size  $256 \times 256$  as a lower resolution results in the loss of pertinent information. Non-Max Suppression (NMS) is used on the predictions to select the most probable regions as shown in Figure. 3.4.

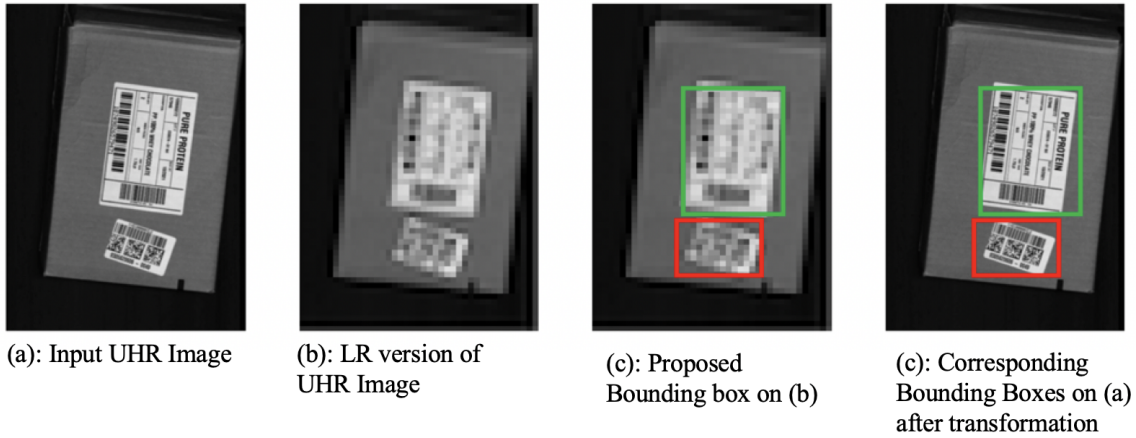


Figure 3.4: Sample region proposal processing phases; (a) UHR Input image; (b) a Low Resolution (LR) of the input is obtained; (c) proposed regions on (b); (d) remapped regions from (b) to (a).

## Y-Net Segmentation Network

As depicted in Figure. 3.6, Y-Net is made out of 3 main modules distributed in 2 branches: a Regular Convolutional Module shown in blue, which constitutes the left branch, and a Pyramid Pooling Module shown in brown, along with a Dilated Convolution Module shown in orange, which, after concatenation and convolution, constitute the right branch.

The **Regular Convolution Module** takes in  $400 \times 400$  output images of the RPN and consists of convolutional and pooling layers. It starts with 64-channel  $3 \times 3$  kernels and doubles the number at each layer. We alternate between convolution and max-pooling until we reach a feature map size of  $25 \times 25$  pixels. This module allows the model to learn general pixel-wise information anywhere in the input image.

The **Dilated Convolution Module** takes advantage of the fact that barcodes have alternating black and white rectangles to learn sparse features in their structure. The motivation for this module comes from the fact that dilated convolution operators play a significant role in the "*algorithme a trous*" for biorthogonal wavelet decomposition (82). Therefore, the discontinuities in alternating patterns and sharp edges in barcodes are more accurately learned by such filters. In addition, they leverage a multiresolution and multiscale decomposition as they allow the kernels to widen their receptive fields with dilation rates from 1 up to 16. Here too, a  $400 \times 400$  input image is used and we maintain 32-channel  $3 \times 3$  kernels throughout the module while the dimensions of the layers are gradually reduced using a stride of 2 until a feature map of  $25 \times 25$  pixels is obtained.

The **Pyramid Pooling Module** allows the model to learn global information about

<sup>1</sup>Our Y-Net architecture resembles the English alphabet letter "Y" and differs from (81) which used a pre-trained encoder network that is augmented with an untrained mirrored network and a decoder network.



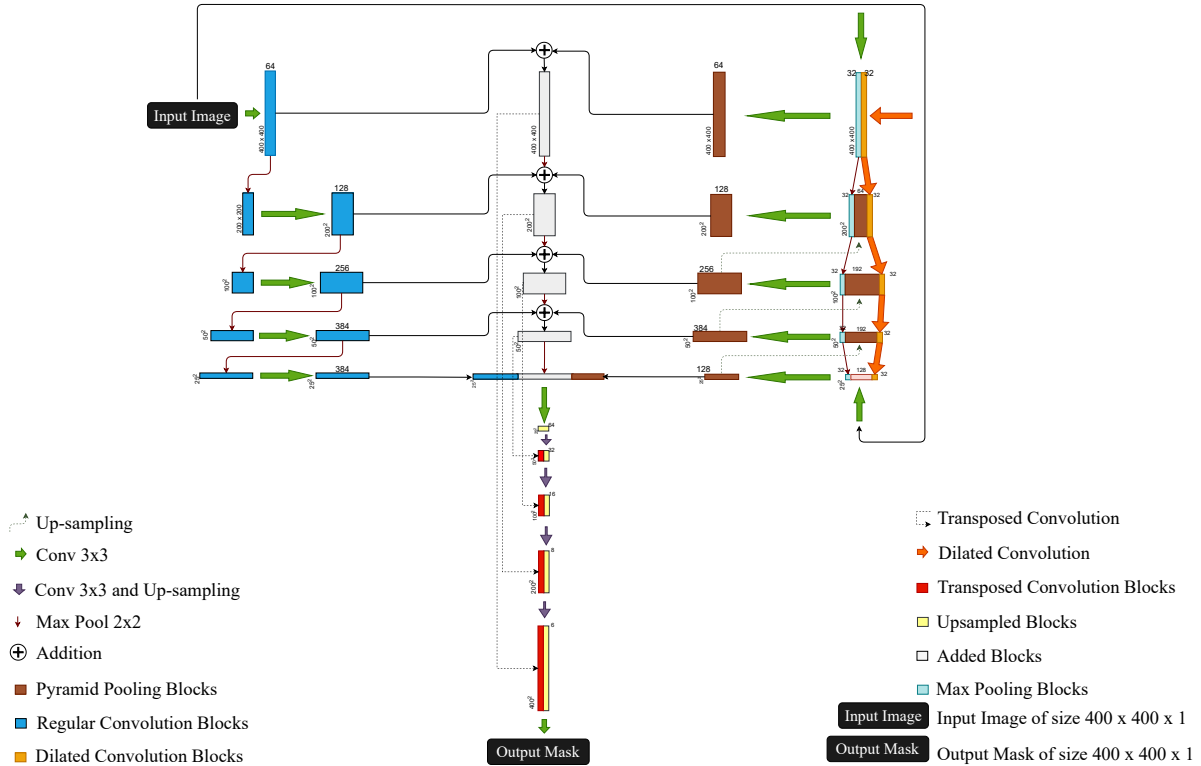


Figure 3.6: Y-Net architecture.

correction for the other, thereby refining the result at each node as shown in white. The nodes are then up-sampled and concatenated with transposed convolution feature maps shown in red and yellow of the corresponding dimension. Throughout the network, we use ReLU as a non-linearity after each layer and add  $L_2$  regularization to account for possible over-fitting scenarios that could have occurred during training. On all datasets, we use 80% for the training set, 10% for the validation set, and the remaining 10 % for the testing set. We use one NVIDIA Tesla V100 GPU for the training process. Since this is a segmentation network and we are interested in classifying background and barcodes, we use binary cross-entropy as loss function. In Appendix B, we explore the use of Y-Net to *Depth* and *Uncertainty* estimation problems and show some qualitative results. We further looked at how they compare to a transformer-based model.

## Bounding Box Extraction

Since some images contain barcodes that are really close to each other, their Y-Net outputs reflect the same configuration, which makes the extraction of individual barcode bounding boxes complex, as shown in Figure. 3.7(a). To separate them effectively, we perform erosion, contour extraction, and bounding box expansion with a pixel correction margin. As shown

	$mAP$ (all)	$AP_{50}$ (all)	$AP_{75}$ (all)	$mAP$ (small)	$mAP$ (medium)	$AR_{50}$ (all)	$AR_{70}$ (all)	$AR_{80}$ (all)	$AR_{90}$ (all)	Latency (ms)	Resolution (px)
Mask R-CNN	.466	.985	.317	.340	.489	.990	.740	.279	.023	94.8	$448 \times 448$
YOLOv4	.882	<b>.990</b>	.989	.815	.897	<b>1.</b>	<b>1.</b>	.995	.873	40.5	$320 \times 320$
<b>Ours</b>	<b>.937</b>	<b>.990</b>	<b>.990</b>	<b>.903</b>	<b>.945</b>	<b>1.</b>	<b>1.</b>	<b>1.</b>	<b>.972</b>	<b>16.0</b>	$400 \times 400$

Table 3.1: Average Precision for Max Detection of 100 and Average Recall for Max Detection of 10 computed using MS COCO API.

	Muenster Dataset				ArTe Lab Dataset			
	DR	Precision	Recall	mIoU	DR	Precision	Recall	mIoU
Creusot et al.	.982	-	-	-	.989	-	-	-
Hansen et al.	.991	-	-	.873	.926	-	-	.816
Namane et al.	.966	-	-	.882	.930	-	-	.860
Zharkov et al.	.980	.777	.990	.842	.989	.814	.995	.819
<b>Ours</b>	<b>1.000</b>	<b>.984</b>	<b>1.000</b>	<b>.921</b>	<b>1.000</b>	<b>.974</b>	<b>1.000</b>	<b>.934</b>

Table 3.2: Mean IoU (mIoU), Precision, Recall, and Detection Rate (DR) at an IoU threshold of 0.5 for Muenster and ArTe-Lab datasets.

in Figure. 3.7(b), the erosion stage allows the algorithm to widen gaps between segmented barcodes that may be separated by 1 or more pixels. The resulting mask is then used to infer individual barcode bounding boxes in the contour extraction stage in Figure. 3.7(c) through border following. A pixel correction margin is used to recover the original bounding boxes' dimensions during the expansion stage, as shown in Figure. 3.7(d). This post-processing stage of our pipeline has an average processing time of 1.5 milliseconds (ms) because it is made of a set of Python matrix operations to efficiently extract bounding boxes from predicted masks.

	Px Acc	Px mIoU	Px Prec	Px Rec
Mask R-CNN	.993	.990	.989	.890
<b>Ours</b>	<b>1.</b>	<b>1.</b>	<b>.999</b>	<b>.999</b>

Table 3.3: Pixel-wise metrics

### 3.3 Datasets and Results

For the synthetic dataset, we use treepoem <sup>2</sup> and random-word <sup>3</sup> to generate UHR and LR barcode images. We use Code 39, Code 93, Code 128, UPC, EAN, PD417, ITF, Data Matrix, AZTEC, and QR, among others. We model the number of barcodes in a given image using a

<sup>2</sup><https://github.com/adamchainz/treepoem>

<sup>3</sup><https://github.com/vaibhavsingh97/random-word>

Poisson process, and a combination of perspective transforms is used to make the barcodes vary in shape and position from one image to the other. We have also added random black blobs at random locations on the original UHR and LR canvases. The real UHR barcodes dataset obtained from Amazon.com, Inc is made of 3.8 million UHR images of resolution up to  $30k \times 30k$  grayscale images and could not be released due to confidentiality reasons. Additionally, the Muenster and Artelab datasets are used with some data augmentation schemes for more samples.

For the RPN, we accumulated the number of bounding boxes inside the proposed regions and divided it by the total number of ground truth bounding boxes. Our implementation yields an accuracy of 98.03% on the synthetic dataset at 10 ms per image and 96.8% on the real dataset at 13 ms per image, while the baseline (79) yields the same accuracies and an average latency of over 2.5 seconds (s) per image for both datasets.

For Y-Net, we use the Microsoft (MS) COCO API, and Pixel-wise metrics to evaluate against (75; 76). By default, the MS COCO API configuration evaluates on *small*, *medium* and *large* areas objects but in our application, the largest detected barcode area is *medium*. Since Y-Net is a segmentation network and does not output confidence scores for each segmented barcode, we propose using *pseudo scores*, the ratio of the total number of nonzero pixels in a predicted mask to the total number of nonzero pixels in the corresponding ground truth mask at the location of a given object.

Table 3.1 shows *mAP* and *mAR* values of the models on the synthetic dataset. As seen, our pipeline outperforms (75), and (76) by a *mAP* of 47.1% and 5.5% and *AP<sub>75</sub>* of 67.3% and 0.1% respectively. Also shown in Table 3.1 is a *mAR<sub>90</sub>* improvement of 94.9% and 9.9% on (75) and, (76), respectively, which highlights that Y-Net continues to yield better *mAR* results even at higher IoU thresholds. Both our approach and (76) achieve an *AR<sub>50</sub>* of 100% and outperform (75) by 1%. For *small* area barcodes, Y-Net outperforms (75) and (76) by a *mAP* of 56.3% and 8.8% and for *medium* area barcodes, Y-Net displays a *mAP* increase of 45.6% and 4.8% on (75) and (76) respectively. In addition, Table 3.3 reveals that Y-Net has much better semantic segmentation performance than (75). Table 3.1 displays that Y-Net performs at least  $2.5\times$  faster than the fastest of models (75) and, (76) on LR images.

Similarly, we have used the Detection Rate (DR), mIoU, Precision, and Recall, as described in (66; 70; 74; 77) on the Arte-Lab and Muenster datasets and as can be seen in Table 3.2, our method outperforms previous works on all of the mentioned metrics. This indicates that our bounding box extraction algorithm is working as expected to detect accurate bounding boxes. However, while it is successful in separating barcodes that are relatively close to each other, it has limitations when barcodes are overlapping, as shown in Fig. 3.7(e). For those occlusion scenarios, the algorithm tends to group the overlapping barcodes into one bounding box instead of separate bounding boxes as shown in Fig. 3.7(f).

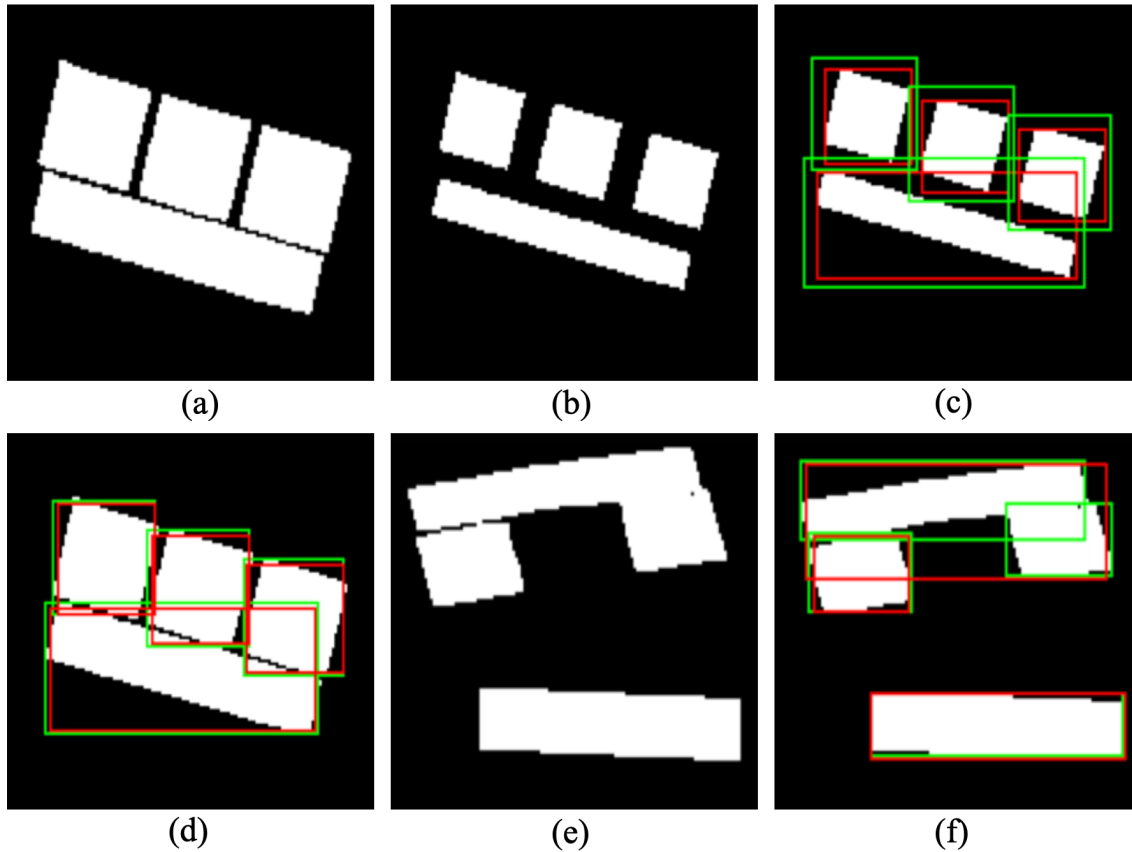


Figure 3.7: (a) Y-Net output; (b) Y-Net output after erosion; (c) extracted bounding boxes –red, ground truth bounding boxes –green on eroded output; (d) final bounding boxes after pixel correction margin on Y-Net output; (e) Y-Net output of occluded barcodes scenarios; (f) final extracted bounding boxes are grouped after pixel correction margin due to overlapping barcodes in the input image.

### 3.4 Discussion

In this chapter, we showed that barcodes can be efficiently, accurately, and speedily detected using Y-Net on UHR images. With *pseudo scores* as *confidence scores*, our approach outperforms existing detection pipelines with a much better latency. In future work, we will be extending this method to the multi-class detection task for small objects in UHR images and videos in a weakly supervised fashion.



## Chapter 4

# Active Perception: Using Non-Auto-Regressive Multimodal Models without Language Instructions via RPTx, a Cross Robot Learning Method with Sensorimotor Pre-training

### 4.1 Introduction

Recent advancements in vision and language processing have ignited significant interest in applying these techniques to pre-training methods for embodied systems in robotics (83; 84; 85). This chapter explores a first key approach of utilizing non-auto-regressive multimodal models in robotics without language, while a second key approach incorporating language in auto-regressive models is explored in the next chapter. In all cases, our goal is to learn good representations that can utilize cross-modal interaction between each element across various modalities.

For non-auto-regressive multimodal models not incorporating language, (3) has shown that self-supervised visual pre-training on large and diverse image datasets is promising on *in-domain* data. However, the complexity of sensory and motor information in robotic data goes beyond what *in-domain* visual pre-training alone can capture. This raises a crucial question: *can we effectively leverage the sensorimotor representations learned from different robotic trajectories to improve the skills of a particular robot?*

To do so, we leverage the extensive and diverse OXE dataset (6) to explore the self-supervised robotic pre-training method proposed by (3), inspired by the masked prediction tasks successfully used in natural language processing (86) and computer vision (87). As



such, the central hypothesis here posits that a robot capable of predicting missing parts of sensorimotor sequences from mixed robot datasets will develop a robust model of the physical world, enhancing its ability to act effectively and adapt easily across different robots.

Given the structure of the OXE dataset, we used RPTx, an adaptation of the RPT model (see Figure 4.1), utilizing a Transformer (88) architecture to manage sequences of sensorimotor tokens. These tokens, derived from camera images, proprioceptive states, and actions, are encoded into sequences where a subset is masked, and the model is trained to predict these masked tokens across various robots and tasks during pre-training. This approach aims to develop cross-modal, spatiotemporal representations that can be generalized across robots and tasks. Following (3), we experimented with high-ratio masking across all modalities and time steps to investigate the learning schemes of complex patterns.

To handle latent vision representations, we employ pre-trained vision encoders (89) to process camera images, making the prediction task more tractable, as demonstrated in (3). We adapted the initial RPT architecture to work with the OXE dataset (6), which includes data from a diverse range of robots, some of which lack certain modalities. To align all inputs within our pipeline, we used learnable tokens for each modality, accommodating scenarios where a modality may be present in one robot’s data but absent in another’s. This adjustment was crucial because, unlike (3), which was trained solely on *in-domain* datasets from a Franka robot with all modalities present and a restricted action space, the OXE dataset includes datasets that may lack essential modalities. Thus, filling in these gaps was necessary to achieve optimal performance.

We utilized over 1.3 million real-world robot trajectories from the OXE dataset (6), encompassing data from various robots and a range of skills in contrast to 1,920 collected *in-domain* trajectories for the pre-training stage. The input modalities at the pretraining stage include sequences of multi-view RGB camera images, proprioceptive robot states, and actions, depending on the specific robot. During the fine-tuning stage, we employed behavioral cloning, focusing on classic robotic tasks such as Picking, Stacking, and Destacking cubes. For these tasks, we collected approximately 550 trajectories for the Picking task, 1,092 trajectories for the Stacking task, and 1,092 trajectories for the Destacking task, all on a Franka robot. Through extensive real-world experiments, we conducted new reliability assessments, identified negative transfer results, and speculated on future work.

## 4.2 Cross Robot Learning with Sensorimotor Pre-training

The pipeline consists of a pre-training stage followed by a fine-tuning stage, designed to enable cross-robot training as outlined above. During the pre-training stage, we modified and adapted the RPT architecture to now accommodate existing and non-existent modalities in the trajectories of various robots. We acquire representations by not only using masked prediction on sequences of camera images, proprioceptive states, and actions, but

## Masked Sensorimotor Prediction

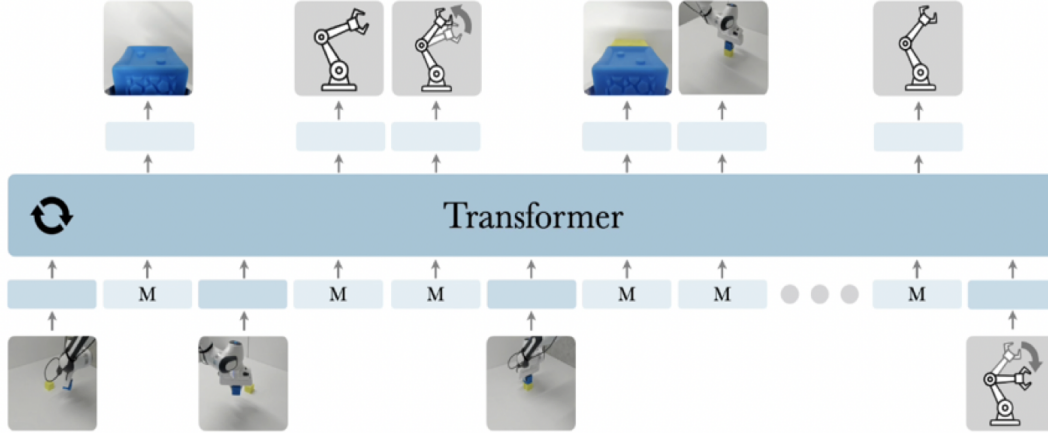


Figure 4.1: Architecture of RPT (3).

also masked-out modalities that may be non-existent in a particular robot trajectory. This stage leverages the diverse data from various robots included in the OXE dataset to attempt to help the model learn generalized representations.

In the fine-tuning stage, these representations are transferred to downstream tasks and refined using behavioral cloning. This process allows the pre-trained model to effectively adapt its learned representations and states to specific robotic tasks and environments, ensuring effective transfer across different robots and tasks.

We reformulate the problem in (3) as follows: Let’s consider  $D$ , the OXE dataset (6), which is a collection of robot-specific datasets  $D_j$  where  $i$  goes from 1 to  $N$  (In our case, we use  $N=54$ ). Each robot-specific dataset  $D_j$  consists of sensorimotor trajectories  $T_{D_j}$  where each trajectory comprises sequences of camera images, proprioceptive states, and actions:  $T_{D_j} = (i_1, s_1, a_1, \dots, i_{T_{D_j}}, s_{T_{D_j}}, a_{T_{D_j}})$ . Importantly, we do not use additional semantic information, such as language instructions or task labels. We do consider the case where a particular robot trajectory will not have some specific modalities since the OXE dataset contains a wide variety of robots and replaced the missing modality in those instances with learnable tokens  $l_j$  resulting in a trajectory  $T_{D_j} = (l_1, s_1, a_1, \dots, l_{T_{D_j}}, s_{T_{D_j}}, a_{T_{D_j}})$  for the case where robot  $j$  is missing some camera images  $i_j$  for example.

We frame the pre-training process as a general masked sensorimotor sequence prediction problem across all modalities and time, as was done in (3). This is implemented through a masked prediction task, inspired by similar techniques in vision and language processing. In this task, a subset of each trajectory described above is masked, and the model is trained to predict the missing content. Given a sensorimotor sequence of the  $j$ th robot dataset in (6),  $L_{D_j}$  tokens, we sample a mask sub-sequence  $M = M_{D_j} \subset [1, L_{D_j}]$  and train the model

to minimize the mean squared error of the masked tokens  $T_{D_j}^M$  conditioned on the observed tokens  $T_{D_j, [1, L_{D_j}] \setminus M_{D_j}}$ .

This general formulation allows us to represent a variety of contextual prediction problems by using different masking patterns. We explore several variants, including random masking at the modality, timestep, and token-level masking. This flexibility is crucial for developing a model capable of adapting to different robots and tasks.

### 4.3 RPTx

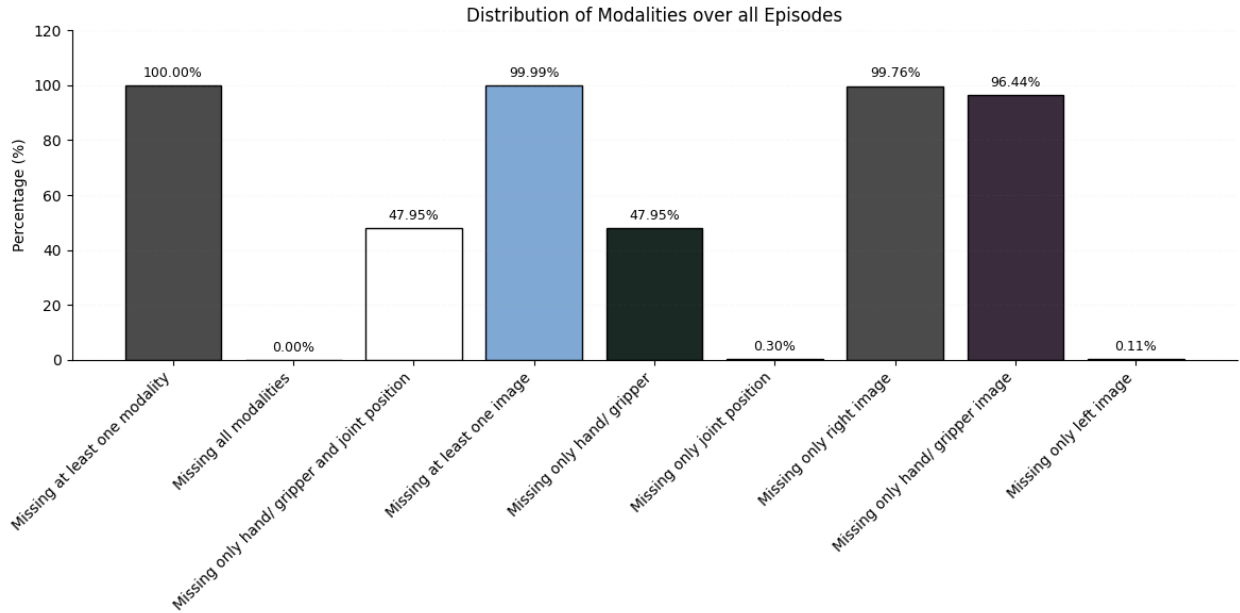


Figure 4.2: Modality Distribution across 54 OXE dataset; no data is missing all modality at once.

We refer interested readers to (3) for details of the RPT architecture. We use the same Vision latent representations, Token encoders, Transformer model, and Prediction heads as in the prior work. Because some of the OXE dataset is missing some modalities, as shown in Figure 4.2, we modified the input of the model to now automatically replace such missing modalities for a given dataset with learnable tokens that were reused across datasets missing the same modality and account for this in the corresponding calculated loss.

## 4.4 Downstream Transfer

In the first section of this chapter, we investigate whether RPTx can learn sensorimotor representations from a diverse pool of robot datasets and transfer knowledge across various downstream tasks and robots. Given that it was shown in (3) that fine-tuning works better than linear probing, we restrict this study to the former. In that setting, we take a pre-trained RPTx model checkpoint and fine-tune it using data specific to a downstream task. This approach aims to leverage pre-training weights as a means to provide a strong initialization to the fine-tuning model.

## 4.5 Discussion

As shown below, our experiments provide strong evidence of negative transfer when using non-auto-regressive multimodal models that do not incorporate large language models (LLMs). We followed with a reliability assessment and speculated on future research directions.

### Negative Transfer Assessment

Negative transfer occurs when a model trained in one context underperforms in a new context due to the transfer of inappropriate or irrelevant knowledge. This phenomenon is particularly significant in active perception, where robots must adapt to varied and dynamic environments. In this section, we discuss the challenges posed by negative transfer and propose strategies to mitigate its impact.

We first observe that errors can emerge in the data preprocessing pipeline. Since the model operates in latent space, it’s crucial to ensure that the image representations obtained from the Vision Transformer (ViT) during both pre-training and fine-tuning align with expectations. Figure 4.3 illustrates our use of Principal Component Analysis (PCA) to investigate why the models initially failed to respond as expected. We discovered that due to a Python version mismatch, the extracted image representations formed three distinct clusters, as shown in Figure 4.3(a), instead of a single, cohesive cluster depicted in Figure 4.3(b). This distribution shift caused the robot to hover during evaluation tasks, rather than moving in the intended direction.

Next, we conducted experiments similar to those in (3), where models were pre-trained on various subsets of the OXE dataset and fine-tuned on the collected Franka dataset. For pre-training, we addressed the dimensionality differences across robots by zero-padding the joint position vector to match the largest dimension of 36-DoF with a default 1-DoF parallel jaw gripper, resulting in a 37-DoF proprioceptive vector. For fine-tuning, we maintained the default 7-DoF joint position with a 1-DoF gripper, controlling the joint position at 10 Hz. Three RGB cameras, as used in (3), were employed for fine-tuning, with one camera attached to the robot’s hand (gripper) and two positioned on the sides.

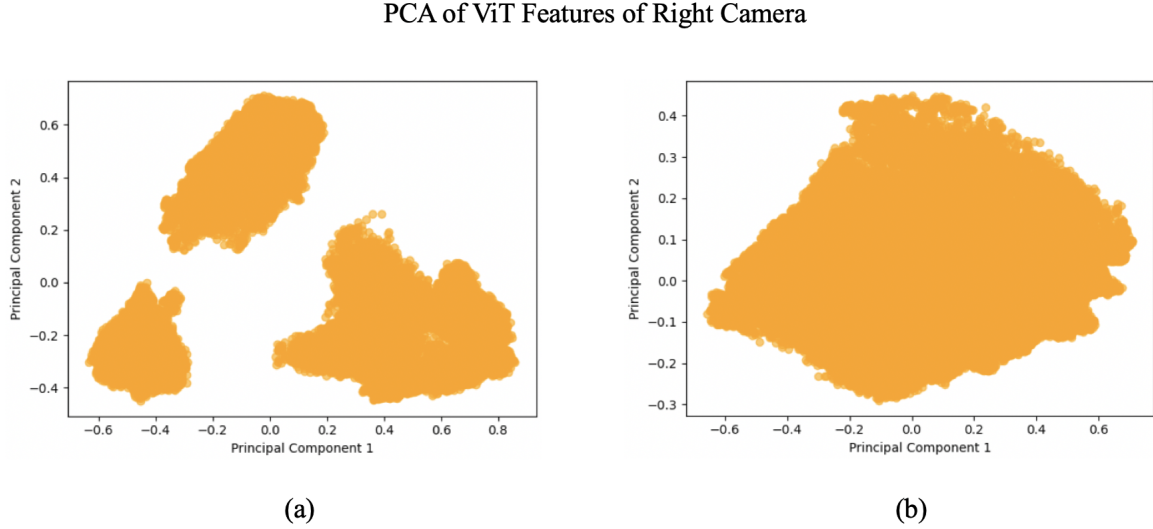


Figure 4.3: PCA of ViT features of the right camera.

We pre-trained the models across various scenarios, including token masking, timestep masking, and modality masking. We also varied the size of the pre-training dataset and explored the impact of context length. Additionally, we evaluated the effects of pre-training exclusively on Franka, xArm, Kuka iiwa, and other non-Franka datasets. For fine-tuning, we focused on the Picking, Destecking, and Stacking tasks, and the fine-tuned models were evaluated across 16 real-world trials, focusing on success rates. All fine-tunings were completed for 900 epochs.

We refer to the model trained from scratch on fine-tuning data as MVP (89), the model pre-trained on *in-domain* dataset as RPT (3), and the RPT model pre-trained on the OXE dataset as RPTx.

- **Pre-training and Fine-tuning Studies on the Picking Task**

We investigated the impact of sensorimotor pre-training on the OXE dataset, followed by fine-tuning for the Picking task, as illustrated in Figure 4.4. Due to the lengthy pre-training process, we applied the pre-training weights from using 100% of the OXE data at epoch 0, after only a few iterations. Token masking was utilized during pre-training, given its superior performance as demonstrated in (3). Our observations indicate that while the general trend of pre-training on *in-domain* data is apparent, RPTx struggled significantly. We attributed the result for the 60 trajectories to the variance in the evaluation process that we documented below in subsequent sections.

- **Pre-training and Fine-tuning Studies on the Destecking Task**

We examined the impact of sensorimotor pre-training on the OXE dataset, followed by fine-tuning for the Destecking task. As in the previous experiment, we used pre-

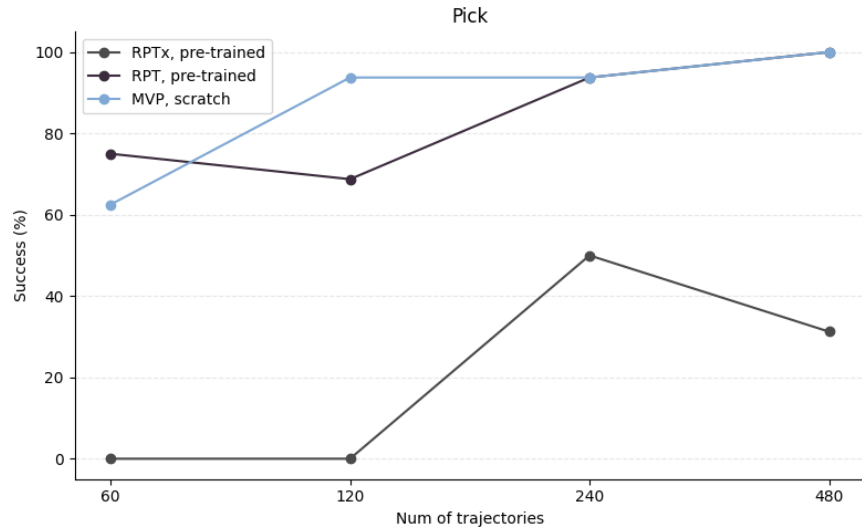


Figure 4.4: Evaluation on the Picking task.

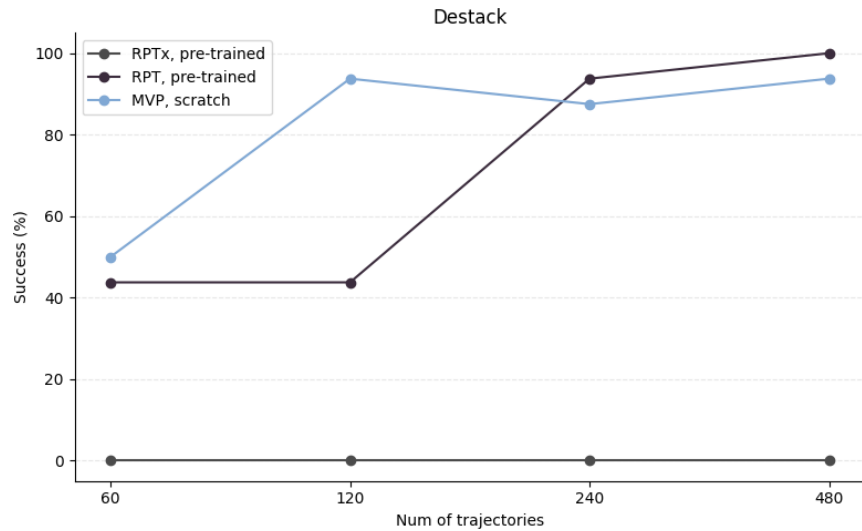


Figure 4.5: Evaluation on the Destack task.

training weights from using 100% of the OXE dataset at epoch 0 after only a few iterations, given the slow pre-training process. Token masking was also employed during pre-training.

We observed a consistent trend between the MVP model and the RPT model for 60 and 120 trajectories. However, for 240 and 480 trajectories, this trend was reversed

by a margin of one success, which we attributed to variance in the evaluation. On the other hand, the RPTx model struggled across all trajectory counts, highlighting the effects of negative transfer as shown on Figure 4.5.

- **Pre-training and Fine-tuning Studies on the Stacking Task**

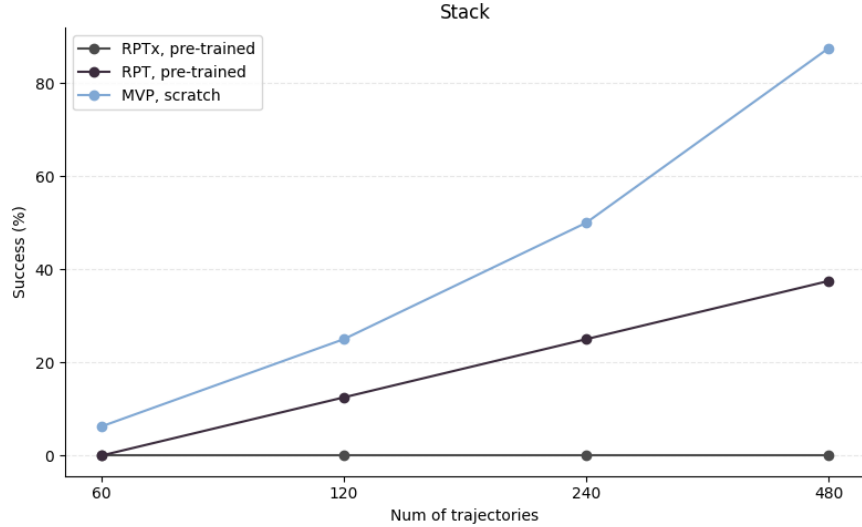


Figure 4.6: Evaluation on the Stacking task.

In Figure 4.6, we analyze the effect of sensorimotor pre-training on the OXE dataset, followed by fine-tuning for the Stacking task, where token masking was also employed during pre-training. The pre-training weights used were obtained from utilizing 100% of the OXE dataset only after a few iterations at epoch 0.

The RPTx model struggled across all trajectory counts, while the MVP model consistently outperformed the RPT model. This result contrasts with the trends observed in the previous two experiments, as the trend is now completely reversed between the MVP and RPT models.

- **Pre-training and Fine-tuning Studies Context Length**

Our goal was to assess the impact of sensorimotor pre-training on the OXE dataset with varying input context lengths: 1, 4, 8, and 16, each consisting of 5 entries (left, hand (gripper), and right cameras, states, actions). We fine-tuned and evaluated the models on the Stacking task. Pre-training weights were applied at epochs 10, 4, 2, and 0 for context lengths 1, 4, 8, and 16, respectively. Due to the lengthy training process, we reduced the number of epochs to only a few iterations. Token masking was employed for all pre-training scenarios.

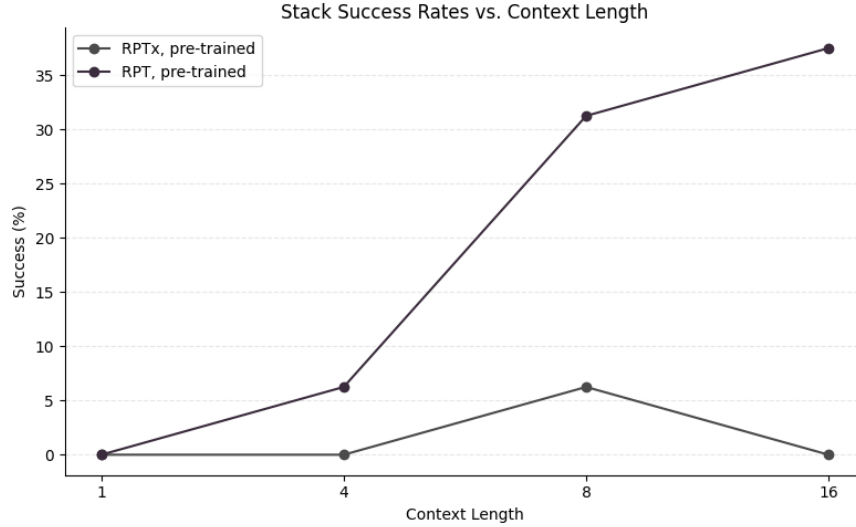


Figure 4.7: Success rate vs. context length.

As shown in Figure 4.7, the RPT model exhibits improved performance with increasing context length. In contrast, the RPTx model shows inconsistent results and often performs near zero. This behavior supports our hypothesis of negative transfer, highlighting the challenges associated with pre-training on the OXE dataset.

#### • Pre-training Data Complexity Studies

We further examined the impact of sensorimotor pre-training on the OXE dataset using 25%, 50%, and 100% of the pre-training data. Each model was fine-tuned and evaluated on the Stacking task, following the same procedure as in previous experiments. Pre-training weights were applied at epochs 6, 2, and 0 (after only a few iterations) for the 25%, 50%, and 100% models, respectively. Token masking was consistently employed throughout the pre-training process.

As shown in Figure 4.8, the RPTx model struggled across all pre-training data sizes, while the RPT model performed better when using the smallest amount of pre-training data, specifically 480 trajectories. These results do not necessarily align with the findings in (3), which suggest that model performance should improve consistently with increased pre-training data size.

#### • Masking Type Studies

We looked at the effect of different masking types (Token, Timestep, and Modality) used during sensorimotor pre-training on the OXE dataset. Each model was fine-tuned and evaluated on the Stacking task, and due to the slow pre-training process, we applied pre-training weights for all three models at epoch 0 after only a few iterations.



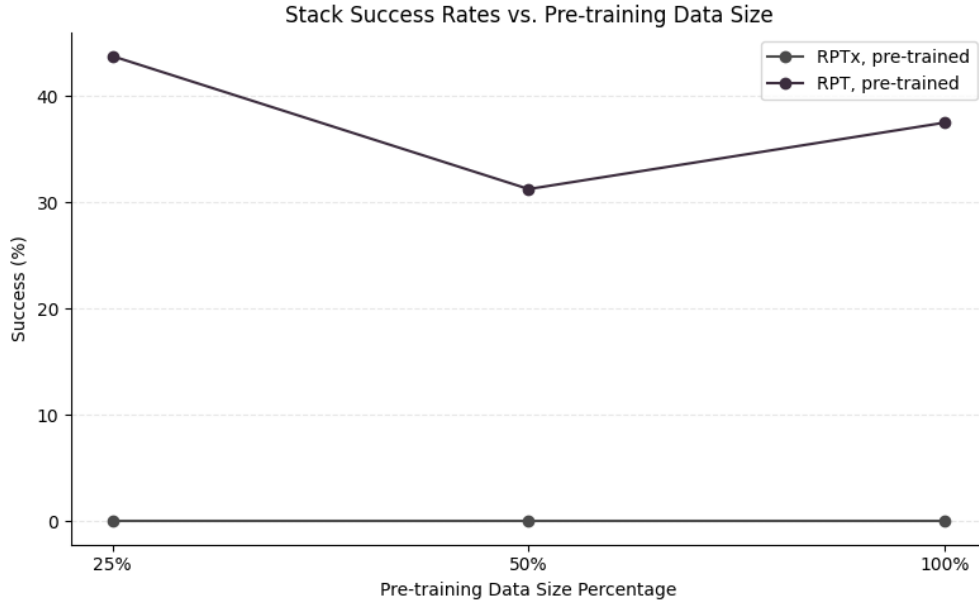


Figure 4.8: Success rate vs. pre-training data size.

From Figure 4.9, the result suggests that the masking scheme to be used for all the experiments above should have been a timestep masking instead of the default token masking used in RPT. It also suggests that an auto-regressive (next-token prediction) style of pretraining, such as that used in scaling language models, may be the key to scaling cross-robot training. Furthermore, it hints that language may be one means through which we can scale cross-robot training. In fact, existing models such as RT-1 (84) and RT-2 (90), which were further expanded to RT-1-X and RT-2-X using the Open-X-Embodiment dataset (6), all utilized language. Unfortunately, these models were not open-sourced at the time of this work. The Octo model (91) also employed language, though it requires improvement in terms of generalization. These observations led us to explore how large language models (LLMs) could be effectively leveraged for robotics, a topic we further discuss in chapter 5.

#### • Pre-training Robot Data Type Studies

We examined in Figure 4.10 the transfer across different robot types, specifically Franka, xArm, Kuka iiwa, and all other non-Franka robots. The models were fine-tuned and evaluated on the Stacking task. Due to the slow pre-training process, we applied pre-training weights at epochs 10, 10, 10, and 3 for the Franka, xArm, Kuka iiwa, and all non-Franka robot datasets, respectively. Token masking was employed throughout the pre-training process.

We observed that for the RPTx model, pre-training on xArm robot data outperformed

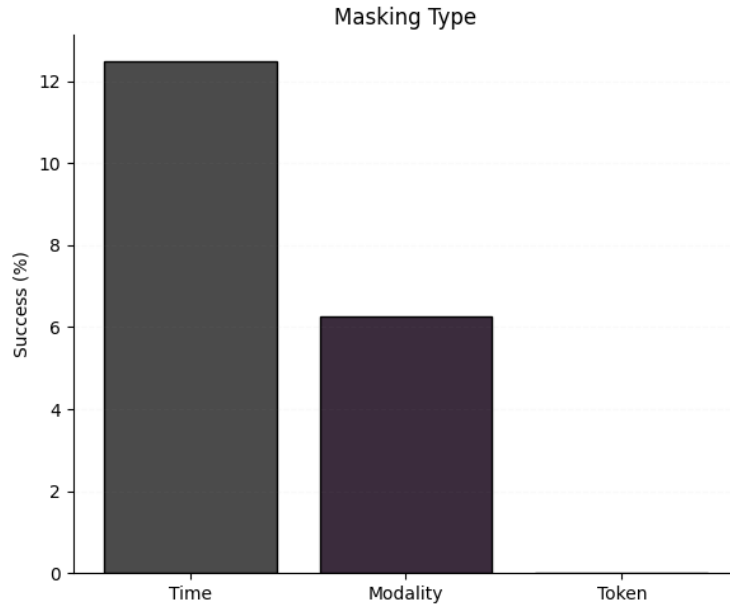


Figure 4.9: Success rate vs. masking type.

pre-training on the other robot types. Conversely, pre-training on Kuka iiwa or all other non-Franka robots showed significant struggles. On the other hand, (3) with the RPT model demonstrates that while sensorimotor pre-training is beneficial, pre-training on the Franka robot dataset performs better than pre-training on xArm data. However, their study only included the Franka and xArm datasets.

## Reliability Assessment

Conducting a reliability assessment is important in active perception as it helps uncover situations where failures can have severe consequences. Traditional methods of assessing the reliability of robotic systems often rely on static benchmarks or simulated environments, which may not fully capture the complexity and variability of real-world conditions. Since we are evaluating the models on a real robot, in this section, we looked at metrics such as *Consistency*, *Recovery Capability*, *Downtime*, and *Operational Safety*.

- **Consistency**

Consistency measures the robot’s ability to repeat tasks under the same conditions and obtain similar outcomes. We used this reliability metric to investigate the variance in performance between different runs of the same experiment.

It is crucial because a robot’s performance should not drastically change from one trial to another under identical conditions. That is, if a robot performs a task successfully

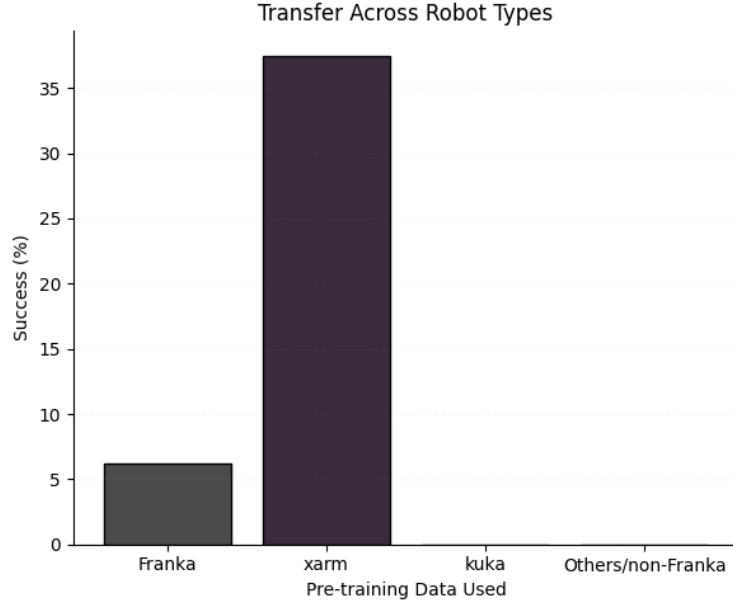


Figure 4.10: Success rate vs. robot type (Cross Robot Transfer).

in one run but fails in subsequent runs, we could assess that there are reliability and consistency issues.

To assess the variance in the evaluation process on the Franka robot, we used RPT models trained from scratch (or fine-tuned from scratch) on collected 60, 120, 240, and 480 trajectories. We then repeated the Stacking experiment multiple times consecutively.

As shown in Figure 4.11, the general trend of model performance improvement with an increasing number of trajectories, as mentioned in RPT, is observed except for the 240-trajectory model, which performed worse than the 120-trajectory model in 3 out of 4 runs. Additionally, there is a significant variance between runs 3 and 4.

We attribute this variance to sensor noise and calibration issues. The system relies heavily on real-time data from all camera feeds and sensed positions, which can introduce noise or errors if the sensors are not well-calibrated. This can significantly impact the model’s ability to perceive and make decisions accurately. Environmental factors, such as lighting, textures, and dynamic elements, can also affect the model’s performance during evaluation. These factors, among others, can compound, leading to large variances in model evaluation, making it challenging to accurately assess and compare their true effectiveness.

To improve consistency across tasks, one solution involves refining the model’s ability to handle variations in environmental conditions and input noise. This could be achieved

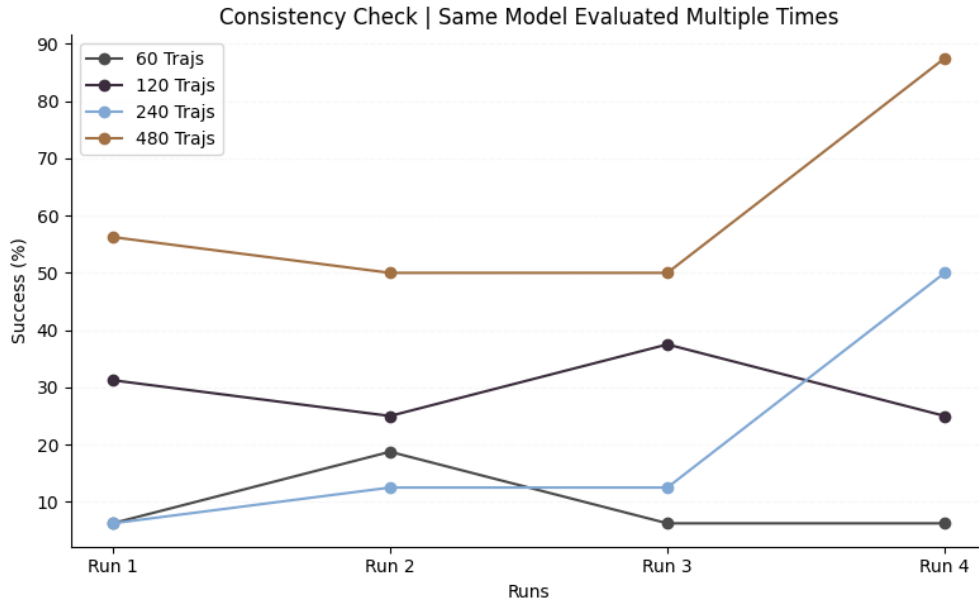


Figure 4.11: Success rate vs. consistency test over several runs.

by incorporating more diverse training data, especially under different lighting conditions, sensor states, and physical disturbances. Techniques like domain randomization and robust data augmentation can expose the model to a wider range of scenarios, allowing it to generalize better and avoid the erratic behavior observed in edge cases. Regular calibration of sensors and actuators will also help mitigate drift over time, leading to more consistent task execution.

#### • Recovery Capability

Recovery capability in robotic systems can be understood as the robot’s ability to recover from minor errors or disturbances while continuing its task without requiring a complete system reset or human intervention. This capability is essential for reliable and resilient robotic operation, especially in real-world environments where unexpected events such as slight object misplacements, sensor noise, or environmental shifts frequently occur.

In our system, recovery is particularly challenging due to the structure of our model at inference time. Since the past trajectory is provided as input, and the model is tasked with predicting actions for the next 16 steps, errors introduced in any of the last steps can propagate through the prediction. This sometimes leads to behavior such as the robot jerking back and forth or even crashing the system when the predicted values fall outside the operational range.

In fact, during certain evaluations, it was observed that once the model produced an out-of-range prediction, the robot consistently experienced a form of oscillation until it failed. This points to the importance of recovery mechanisms that prevent small errors from compounding and causing total task failure. This behavior was also observed during the data collection process.

A highly reliable robotic system should not only perform tasks accurately but also exhibit robustness against such minor disturbances. The ability to adapt and recover without a full reset is crucial for autonomy, especially in scenarios where human intervention is impractical. High recovery capability, therefore, directly enhances a robot's operational resilience and increases its utility in more complex, unstructured environments.

During our experiments, recovery capability was evaluated by deliberately introducing small perturbations during task execution to observe how the robot would respond. These perturbations included slight shifts in the environment, such as minor changes in lighting conditions or changing the position of the cube after initialization. Across all models, a consistent sensitivity to such small environmental changes was observed, highlighting the need for improvements in the system's robustness and the design of recovery mechanisms to ensure continued performance in the presence of external disturbances.

Enhancing recovery capability will require equipping the robot with adaptive control algorithms that can recognize and respond to minor errors before they escalate. By integrating a feedback loop that continuously monitors the robot's performance, the system could detect deviations in real time and make corrective adjustments. Techniques like reinforcement learning or adaptive neural networks can enable the robot to learn recovery strategies from its own failures, improving autonomy. Additionally, building fault-tolerant mechanisms such as graceful degradation will help the system recover from errors gracefully without requiring external intervention.

- **Downtime**

Downtime, in our context, is intentionally structured to provide the robot with periods of rest, which diverges from the traditional focus on minimizing non-functional time. Rather than viewing downtime solely as a loss of productivity, we integrate it as a proactive measure to enhance the robot's long-term reliability and performance, particularly in continuous operations. Allowing the robot to rest and recalibrate periodically mitigates the effects of sensor drift, motor fatigue, and the accumulation of small errors that could otherwise lead to performance degradation.

Instead of perceiving downtime as a detriment, we consider it a crucial component of system maintenance and health management. Through careful monitoring, we found that by strategically increasing downtime, we could significantly improve the robot's consistency and reduce the likelihood of system failures during extended evaluations.

This approach shows the importance of downtime as a tool not only for preserving the system’s functionality but also for improving its overall resilience.

During our evaluations, we observed that after prolonged operation without breaks, the robot’s performance deteriorated significantly even on the MVP and RPT models. This prompted us to explore maximizing downtime as a deliberate strategy to ”rest” the robot between tasks. Counterintuitive as it may seem, this method proved beneficial, leading to more stable and reliable performance over time. The brief periods of inactivity allowed the system to recover, resulting in improved evaluation outcomes.

This balance between active operation and controlled downtime helps prevent system overstrain and failures, turning downtime management from a maintenance necessity into a performance optimization technique. By incorporating well-timed recovery intervals, we can bolster the robot’s long-term operational reliability and sustain higher levels of performance.

To manage downtime, introducing scheduled maintenance periods for the robot is key. This includes periodic system checks and recalibration intervals to address issues like sensor drift and motor fatigue, which can accumulate over extended use. Implementing self-diagnostic tools would allow the robot to detect when a recalibration or rest is needed, minimizing downtime by conducting these operations autonomously when it is least disruptive. By balancing active task execution with structured downtime, we can maintain the system’s reliability and avoid performance degradation due to overuse.

#### • Operational Safety

Operational safety refers to the robot’s capacity to perform tasks without causing harm to itself, its surroundings, or humans in its operating environment. Given that robots often operate in close proximity to people or fragile objects, maintaining a high standard of safety is critical. A reliable robotic system must adhere to safety protocols, ensuring it avoids collisions, prevents excessive force during manipulation, and accurately interprets its surroundings to minimize risks.

During our experiments, operational safety was assessed by observing the robot’s behavior while interacting with objects and its environment. In the Stacking task, for example, we evaluated how safely the robot handled objects, testing its ability to avoid dropping them or applying unnecessary force. Models with higher error rates or inconsistencies were more likely to pose safety risks, particularly in edge cases where sensor misinterpretation could lead to erratic or unintended movements.

We observed several concerning behaviors during these experiments. On some occasions, the robot dropped objects in mid-air, seemingly at random. In other instances, the gripper missed the target entirely, bypassing the object and colliding with the table instead. There were also moments when the robot exhibited uncontrollable jerking motions, leading to system crashes. These issues highlighted the need for improved

safety measures, as unpredictable actions not only jeopardized task success but also increased the risk of damaging the system or its environment.

Ensuring robust operational safety is essential, particularly as robots are integrated into environments shared with humans and delicate objects. By identifying and addressing these safety lapses, we aim to enhance the system’s operational safety and minimize potential hazards during real-world deployments.

Improving operational safety involves developing more sophisticated collision avoidance and force-limiting protocols. This could be achieved through the use of advanced sensors, such as high-fidelity force-torque sensors and overhead vision systems, which provide the robot with more precise awareness of its environment. Combining these with Machine learning models that predict human movement and object trajectories can further improve interaction safety as it could anticipate a dangerous movement and shut down the system. Implementing hard limits on force and motion speed will also help ensure that, in the event of an unexpected behavior, the robot poses minimal risk to humans or objects. Regular safety audits and testing under edge cases can further identify vulnerabilities and refine protocols to prevent system crashes or erratic movements.

## Future Directions and Speculation

One potential way to improve the RPTx system’s performance would be to run the pre-training step for significantly longer than the few iterations we have experimented with. However, this is costly and can take months to converge on existing infrastructures at the time of this work. Another way that could potentially work would be to use Reinforcement Learning (RL) methods instead of the Behavioral Cloning (BC) method on the downstream tasks to improve learning, as has been explored by (92; 93; 94; 95). By allowing robots to learn from their experiences in real time, these techniques can improve the adaptability and robustness of the systems. For instance, a robot using reinforcement learning could continuously refine its perception model based on feedback from the environment (unstructured or dynamic), which will lead to a more accurate and reliable performance over time while using imitation learning alone on the same robot will be insufficient.

Another exciting direction to overcome this issue is the integration of auto-regressive large language models (LLMs) with active perception systems as has been explored by (84; 96; 97; 98). LLMs have not only been shown to have the potential to enhance the decision-making capabilities of robots by enabling them to interpret and generate complex actions via instructions input based on natural languages, but also seem to facilitate cross-robot transfer when not trained solely on *in-domain* data. For example, a robot equipped with an LLM could understand and execute high-level commands, such as ”navigate to the kitchen and find the red cup,” by integrating visual perception and actions with linguistic context. This opens the door for more intuitive human-robot interaction, reducing the need for highly specific training for different tasks.

By aligning linguistic, visual, and action space, robots can now achieve a more holistic understanding of their surroundings, improving their ability to make context and environment-aware decisions. In chapter 5, we discuss our findings on exploring such an approach in robotics via instruction tuning.



## Chapter 5

# Active Perception: Using Auto-Regressive Multimodal Models with Language Instructions via LLARVA, a Vision-Action Instruction Tuning Method that Enhances Robot Learning

### 5.1 Introduction

In this chapter, which contains material from our work in (4), we shift focus to incorporating language into auto-regressive multimodal models. Recent advancements in instruction-tuned Large Multimodal Models (LMMs) have achieved impressive results in image captioning and visual question answering. Despite these successes, applying LMMs to robotics remains challenging, with many models struggling to generalize across diverse robotic scenarios despite extensive training.

We show that integrating language with robotics through a new instruction-tuning technique that utilizes structured prompts to unify various robotic tasks and environments can significantly enhance performance. By leveraging auto-regressive large language models (LLMs), we aim not only to overcome the negative transfer experienced in the previous chapter with RPTx but also to improve generalization across diverse robotic settings. As detailed in (4), we used 8.5 million images from the Open X-Embodiment dataset and evaluated the model’s performance on a real Franka robot. Our findings suggest that this approach provides robust performance improvements compared to non-auto-regressive attempts that do not incorporate language like RPTx. This chapter highlights how the integration of language data and the utilization of auto-regressive methods can address challenges in robotic

applications and facilitate better generalization across different tasks and environments. In Appendix C, we give more details about the dataset used, and we elaborate on the steps taken to successfully train these models on High-Performance Computers (HPC).

## 5.2 LLARVA

Recent developments in instruction-tuned Large Multimodal Models (LMMs), such as InstructBLIP (99), InstructGPT (100), LLaVA (101; 102), and PALM (97), have led to significant advancements in tasks that combine vision and language. Despite their success in these areas, similar models applied to robotics often face different levels of challenges in achieving consistent and effective results across various practical environments (91; 96; 103; 104). These difficulties may arise from the unique conditions in robotics, including the variability of real-world settings, differences among robots, and the need for precise action control. Given that multimodal instruction tuning has proven effective for LMMs in other domains, it is reasonable to explore its potential in robotics as well. In this section, we discuss a method for instruction tuning that focuses on vision and action, aiming to align a language model’s primary pre-training task of predicting subsequent words with the demands of different robotics applications.

In particular, we discuss, *LLARVA: Large Language model for Robotic Vision and Action* (4), an open-source, instruction-tuned auto-regressive large multimodal model designed specifically for robotics. The model is engineered to adapt efficiently across different environments and robot setups. This approach involves creating a unique instruction prompt that combines information about the robot type, task, scene, and control methods into a natural language format that is compatible with current LMMs. As shown in Figure 6.1, we outline a crafted instruction tuning method for robotics: the model receives a description that includes details about the robot, its control mode, the task at hand, and sensory input, and it is tasked with forecasting future actions based on this description thereby allowing the system to utilize language prompts as a common framework for active perception.

Because aligning vision and action in robotics remains complex, many models struggle with 3-D representations like voxels or point clouds due to compatibility issues with existing LMMs, which typically process only single images and language inputs. To address this, we use 2-D images, which integrate better with LMMs. Our approach involves predicting “visual traces,” 2-D projections of an end-effector, alongside subsequent robot actions to improve alignment between vision and action. By leveraging the Open X-Embodiment dataset (OXE) (6) for generating these traces and structuring our instructions, we enhance action prediction accuracy. Our evaluations demonstrate that this method allows *LLARVA* to generalize effectively across various robotic tasks and environments, showing competitive performance against current baselines in both simulated and real-world settings.

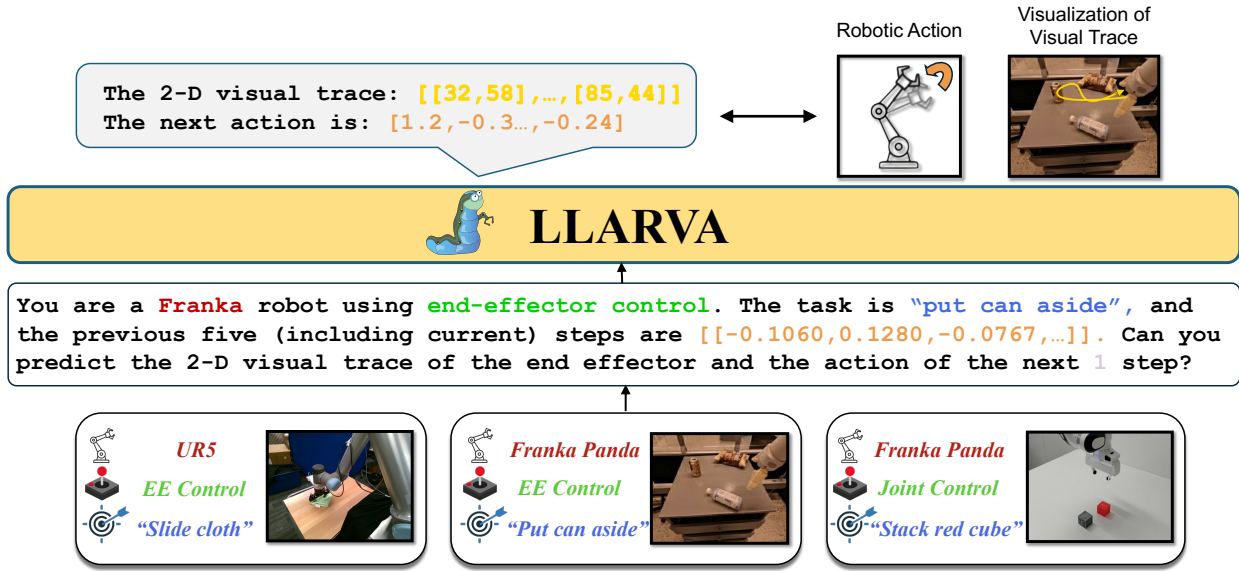


Figure 5.1: Overview of LLARVA (4). We used language instruction that contains *robot model*, *control mode*, *robot task*, *proprioceptive information*, and *number of predicted steps*, and outputs text with the next *robot action(s)* and the *visual trace* for the remainder of the episode.

## 5.3 Method

We refer interested readers to (4) for extensive details of the architecture and summarize the key points here.

### • Input

There are two main modalities to the inputs, which are the visual observations and the language instructions. The visual component,  $o_t$ , is an image showing the environment at a specific time  $t$ . The language input,  $l_t$ , provides detailed guidance by including information about the robot type  $\mathcal{R}$  (e.g., Franka, UR5, xArm), control mode  $\mathcal{M}$  (e.g., joint or end-effector control, absolute or delta control), task  $\mathcal{I}$  (e.g., “open the drawer”), and previous sensory data  $\mathcal{S}$  (e.g., positions or velocities), as well as a request to forecast future  $n$  actions and it is framed as a prompt shown below that asks the model to predict the end-effector’s trajectory and subsequent actions.

$l_t$  = “You are a  $[\mathcal{R}]$  robot using  $[\mathcal{M}]$  control. The task is  $[\mathcal{I}]$ , and the previous  $[h]$  steps are  $[\mathcal{S}]$ . Can you predict the trajectory of the end-effector and the action of the next  $[n]$  steps?”

To handle tasks with varying durations effectively, the model also accommodates flexible proprioceptive inputs. These inputs are presented as a sequence of past joint positions and gripper states,  $\mathcal{S} = s_{t-h}$ , where  $h$  indicates the number of previous steps

considered. This approach ensures that the model can be robustly trained for both short-term and long-term tasks, adapting to different time horizons as needed.

- **Architecture**

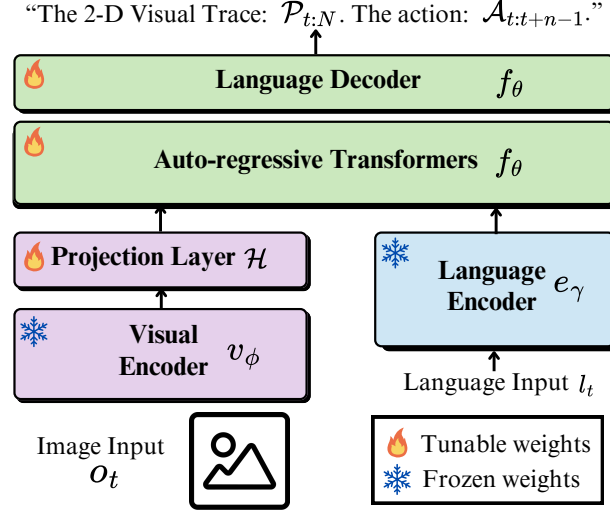


Figure 5.2: LLARVA architecture (4).

The LLARVA architecture, shown in Figure 5.2, integrates current visual data  $o_t$  with corresponding language instructions  $l_t$  and predicts both the sequence of actions for the upcoming  $n$  steps, denoted as  $\mathcal{A}_{t+n-1}$ , and the 2-D visual traces  $\mathcal{P}t$  for the entire episode. The model is represented by:

$$\pi(o_t, l_t) \rightarrow \mathcal{A}_{t+n-1}, \mathcal{P}t \quad (5.1)$$

In this framework, the visual input is processed through a pre-trained CLIP ViT- L/14 (105) vision encoder  $v_\phi(\cdot)$ , which extracts features and maps them into a latent space using an MLP layer  $\mathcal{H}$ . These features are then aligned with the language tokens generated by the Llama 2 (7B-parameters) (106) language encoder. The combined visual and language tokens are fed into the auto-regressive transformers used in Large Language Models of the LMM  $f_\theta$ , which are designed to predict the next tokens in such a way that:  $T = f_\theta(v_\phi(o), e_\gamma(l))$ .

- **Outputs**

- **Visual Traces** We found that predicting Visual Traces in the process helps improve the alignment between visual inputs and robotic actions, enhancing accuracy. These 2-D Visual Traces are sequences of  $(x, y)$  coordinates tracking the gripper’s path and are aligned with the image  $o_t$  at each time step:

$$\mathcal{P}_t = (x_i, y_i) \mid i = t, t + 1, \dots, N \quad (5.2)$$

Here,  $(x_i, y_i)$  denotes the  $i$ -th coordinate in the entire visual trace for the episode, and  $N$  represents the number of time steps in the episode

- **Robot Actions** Robot Actions are obtained by leveraging the language model decoders, which transform these multimodal inputs into actionable outputs, thereby utilizing a unified vision-action space to generate effective robotic responses.

### • Training

While keeping the vision encoder and language encoder frozen, we use instruction tuning to train the auto-regressive transformers using standard LoRA adapters (107) for both the pre-training and fine-tuning stages. Each image  $o_t$  for an episode is accompanied by a language instruction  $l_t$ , and the ground-truth annotations consist of robotic actions  $\hat{A}_{t:t+n-1}$  and visual traces  $\hat{\mathcal{P}}_{t:N}$ . Next, given  $o_t$  and  $l_t$ , we predict the next actions and 2-D visual trace. Specifically, for a response  $R$ , we compute the probability of the target actions and target visual traces by the following equation:

$$p(\hat{A}_{t:t+n-1}, \hat{\mathcal{P}}_{t:N} \mid o_t, l_t) = \prod_{i=1}^{|R|} p_{\theta}(x_i \mid o_t, l_t) \quad (5.3)$$

where  $\theta$  represents the trainable parameters,  $x_i$  is the current prediction token, and  $n \leq N$ . To calculate loss, we use the standard cross-entropy function with these probabilities. Next, we describe our two-step training process, the large-scale pre-training and the fine-tuning for a downstream task.

- **Step 1: Vision-Action Instruction Pre-training** We begin with an LMM that has been pre-trained on vision-language (VL) tasks. In order to generalize across robotic tasks, scenarios, and environments, the model is pre-trained on our large-scale vision-action instruction dataset. Due to the diversity of this dataset, our model is trained simultaneously for multiple configurations of prompt variables such as robot type  $\mathcal{R}$ , control mode  $\mathcal{M}$  or task instruction  $\mathcal{I}$ . Using language as input allows us to bridge fundamental gaps between subsets brought by these different configurations. This extensive and varied training process can establish a powerful LMM framework that can be further fine-tuned and adapted to handle various robotic settings. We note that this pre-training stage is different from standard LMM pre-training. As opposed to aligning the two modalities using a projector in VL, here we align the two modalities for generalizing robotic configurations.
- **Step 2: Fine-tuning for Downstream Tasks** Unlike other fields, a robotic model must be fine-tuned on a downstream task before it can be evaluated due to the practical considerations of real-world physical properties. Therefore, we

fine-tune the pre-trained model using a small dataset with a fixed configuration for the factors defined in the constructed prompt (e.g., the instruction has the same robot type  $\mathcal{R}$ , control mode  $\mathcal{M}$ , etc.). Having seen diverse data samples makes it easy for the model to adapt to specific downstream settings resembling what it has already encountered in pre-training.

- **LLARVA Dataset** In order to pre-train LLARVA, we generate 8.5M image-visual trace pairs from the Open X-Embodiment (OXE) dataset (6). As shown in Figure C.2, our dataset consists of images from a diverse collection of 37 OXE subsets with 13 different robots, including a wide assortment of tasks, environments, cameras (and thus images), and end-effectors, among other factors. For each image in an episode, we calculate the 2-D visual trace of the end-effector  $\mathcal{P}_{t:N}$ . For this purpose, we use a bounding box detector (108) that is trained specifically on each of the different end-effectors in OXE. The center points of bounding boxes are used for a simpler representation, and the visual trace for step  $t$  is then the ordered list of all center points from image  $t$  to image  $N$ .

During fine-tuning, we use a comparable number of episodes as (3) for each task, both with/without visual trace joint training. We condition on joint positions of the previous 16 steps, and predict and execute 7-dimensional delta joint positions and 1-dimensional gripper status for the following 16 steps. We take 16 episodes to evaluate RPT and LLARVA, 10 episodes for Octo, averaging 5 times to get the final success rate. For fine-tuning data collection and evaluations, we use a real Franka Emika Panda robot with a Franka gripper. A Logitech BRIO 4K camera positioned to the right of the Franka robot provides single-view RGB (without depth data) vision input to our model. Camera autofocus is disabled, and the data is captured at 640x480 resolution. The model inference is done on a 48GB NVIDIA A6000. We use the data collection code and process from <https://github.com/Max-Fu/franka-scripted> to collect data for picking, stacking, and destacking tasks. The script generates for each of the arbitrary number of episodes, x-y positions on the table plane using a uniform random distribution for each axis. The script directs the robot to place the cube at each location and then collects the camera and joint information as the robot is directed to pick, stack, or destack the cubes. Vision is not used during this process as the cube locations are all generated and therefore known.

For training and execution on the collected Franka Emika Panda robot data, we start with our LLARVA model that has undergone vision-action instruction pre-training on OXE as described in step 1 above, and perform step 2 for four epochs on 1920 episodes of task-specific downstream data (e.g., picking, stacking, destacking) using 8 A100 GPUs. This is similar to other baselines, such as RPT (3), which uses an equal amount of *in-domain* episodes (1920) for pre-training, with an additional 120-240-480 episodes used for various fine-tuning tasks and experiments. Additionally, (3) uses

three camera views for each episode, while LLARVA uses only one. More details of the dataset are reported in (4) and are shown in Appendix D.

## 5.4 Discussion

Method	2-D Visual Trace	Pick	Stack	Destack
RPT	-	87.50	31.25	93.75
RPTx	-	0.0	0.0	0.0
Octo	-	60.00	10.00	40.00
LLARVA	✗	81.25	50.00	87.50
LLARVA	✓	<b>93.75</b>	<b>56.25</b>	<b>100</b>

Table 5.1: Success rate (%) of LLARVA on the Frank robot. We compare LLARVA with RPT, RPTx (480 trajectories), and Octo by taking each pre-trained model and fine-tuning it on the same set of demonstrations. LLARVA outperforms the others on all the tasks.

Table 5.1 presents a comparison of success rates between *in-domain* results of the non-auto-regressive models, which do not utilize language like RPT (3) and RPTx, non-auto-regressive models that do incorporate language, such as Octo (91), and auto-regressive models that do incorporate language like LLARVA (4). We omitted the results of RPTx in this discussion because of the observed negative transfer mentioned above, where all evaluated tasks for the 480 trajectories returned zeros. This comparison covers various robotic tasks, including Picking, Stacking, and Destacking, offering valuable insights into how incorporating language frameworks can enhance performance and address negative transfer issues observed in RPTx.

RPT (3) and Octo (91) serve as baseline methods for this analysis. RPT demonstrates high success in the Destacking task (93.75%) but shows relatively lower performance in Picking (87.50%) and Stacking (31.25%) tasks. In contrast, Octo exhibits improved performance in Picking (60.00%) but falls short in Stacking (10.00%) and Destacking (40.00%).

LLARVA, on the other hand, outperforms both RPT and Octo across all tasks. Even without using 2-D Visual traces, LLARVA achieves competitive results (81.25%) in Picking, 50.00% in Stacking, and 87.50% in Destacking. When 2-D Visual traces are included, LLARVA’s performance markedly improves, achieving 93.75% in Picking, 56.25% in Stacking, and a perfect 100% in Destacking.

These results suggest that LLARVA offers superior transferability, versatility, and robustness compared to RPT and Octo. The significant performance boost observed with the inclusion of visual traces indicates that further refinements within LLARVA could lead to even greater improvements. While RPTx experienced severe negative transfer, *in-domain* RPT excelled in Destacking, and Octo performed relatively better in Picking. This reinforces the benefits of incorporating language in auto-regressive large multimodal models for robotics. That is, LLARVA’s balanced and high performance across all tasks presents the

effectiveness of multimodal models with language instruction, showcasing their ability to handle diverse tasks effectively.

In summary, these results highlight the potential of fine-tuning and adapting instruction-tuned models to enhance robotics performance. The *in-domain* data-driven and non-auto-regressive RPT provides a valuable benchmark for LMMs without language instructions, while Octo offers insights for those with non-auto-regressive models with language model integrations. However, the substantial improvements achieved by LLARVA demonstrate the clear advantages of advanced language-driven auto-regressive models and techniques in optimizing robotic task execution.

## 5.5 Future Work

The results suggest several promising avenues for future research and development. One key area is to dig deeper into the specific enhancements or modifications within LLARVA that contribute to its superior performance when using visual traces. Understanding these factors could offer valuable insights for further refining LLARVA. Historically, advancements in this area have explored a variety of techniques, from foundational image recognition methods (109; 110; 111) to object-focused video analysis approaches (112; 113; 114; 115). These techniques have included object tracking, interaction modeling, and scene graph generation (116; 117; 118). Recently, there has been increased interest in combining vision with language for tasks like referring expression localization (119; 120; 121) and text-guided semantic segmentation (122; 123). However, none of these studies have explicitly investigated why and how modeling the end-effector location with a language model could lead to such performance enhancements. It will therefore be beneficial to look deeper to understand what is going on.

While LLARVA performs better on the Franka robot, it is worth mentioning that it takes too long to execute (8 minutes). In future work, significantly improving the latency as discussed in chapter 3 will be beneficial.

Additionally, exploring how LLARVA scales under various conditions or with different real-world tasks could further validate its robustness and adaptability. Testing LLARVA in diverse environments and across a broader range of tasks would provide deeper insights into its practical applicability and effectiveness.

In conclusion, the observations presented highlight the significant advantages of using language-driven auto-regressive Large Multimodal Models (LMMs) for robotics. LLARVA stands out as a promising solution for enhancing robotic task performance, demonstrating considerable potential for future advancements and applications. The continued exploration of LLARVA’s capabilities and integration with other modalities could lead to substantial improvements in robotic systems and broaden their scope of use.



## Chapter 6

# Active Perception: Using Auto-Regressive Multimodal Models for Geospatial Reasoning Segmentation via LISAt, a Language-Instructed Segmentation Assistant for Satellite Imagery

### 6.1 Introduction

Active perception, as mentioned in previous chapters, requires learning representations that enable systems to interpret sensory data in a goal-directed and context-aware manner. Just as robots must perceive and act using multimodal inputs and language, remote-sensing systems increasingly need similar capabilities. These systems must go beyond static object detection to reason about complex queries grounded in visual context. This chapter, which includes material from our work (124), explores how reasoning segmentation, where models generate segmentation masks from natural language queries, is a step toward active perception and tasking in the geospatial domain.

Segmentation models for remote-sensing have been a staple of geospatial analysis, supporting applications ranging from disaster response, environmental monitoring, and more (125; 126). These models typically operate within rigid boundaries but struggle to adapt to real-world scenarios in which the ability to segment regions based on flexible, user-defined queries, which are tasks often referred to as reasoning segmentation, is paramount (127). For instance, a query such as “identify flood-prone urban areas” or “which regions have observed urban expansion” demands that segmentation models move beyond static object recognition and into contextual, task-specific reasoning. Despite the clear need for such capabilities, rea-

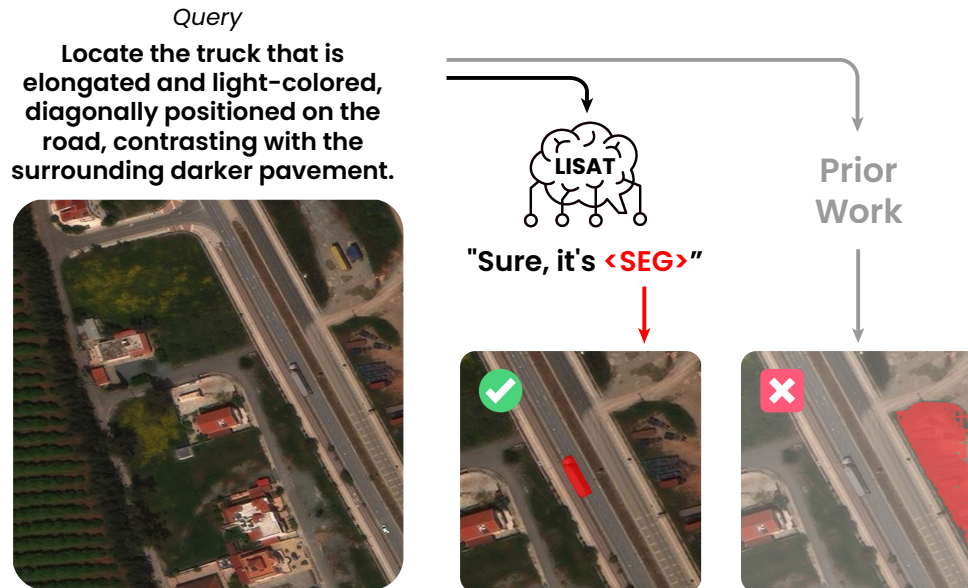


Figure 6.1: Existing models struggle to generate accurate segmentation masks for complex natural language queries in remote-sensing imagery. LISAT, our open-source, open-data, foundation model for geospatial reasoning segmentation trained on GRES, our new semi-synthetic dataset for remote-sensing reasoning segmentation, helps to bridge the gap between State-of-the-Art (SOTA) reasoning segmentation models and remote-sensing domains.

soning segmentation remains largely underexplored in the remote-sensing domain, limiting the adaptability of segmentation systems in practice.

Adapting existing vision models for the remote-sensing domain introduces a unique set of challenges differing fundamentally from those encountered in natural imagery (128). Remote-sensing images can drastically vary in terms of clutter, objects of interest can be very small or very large, and implicit interactions between objects can span over long distances. Additionally, remote-sensing is characterized by both subtle visual differences between drastically different types of objects, where it is challenging to distinguish between objects that look similar in satellite images but have vastly different semantic meanings (e.g., small cars vs. buildings), and extreme variations in scale along with object diversity, where remote-sensing images contain objects of vastly different sizes (e.g., entire cities, forests vs. individual trees). Compounding these difficulties is the lack of high-quality annotated data consisting of natural language queries and remote-sensing imagery pairs. As a result, models trained on natural image datasets or designed for general-purpose reasoning struggle to achieve high performance when directly applied to geospatial tasks (129; 130).

Recent geospatial-specific foundation models, such as RS-GPT4V (129), EarthGPT (130), and others (131; 132) have demonstrated strong performance in related visual understanding and reasoning tasks such as visual captioning and visual question answering. Despite

these advances, such models remain limited to textual outputs and cannot generate segmentation masks or localize objects within images. This lack of segmentation capability presents a significant barrier for applications that require spatially explicit reasoning. Some vision-language models can generate segmentations from text queries (127; 133; 134), but they struggle to adapt to the unique challenges of remote-sensing, mentioned above (see Table 6.2).

We address these challenges by introducing LISAT (**L**anguage **I**nstruction **S**egmentation **A**ssistant for **S**atellite **I**mages), an open-source and open-data vision-language model that bridges the gap between reasoning segmentation and remote-sensing foundation models. Central to LISAT’s development are two new datasets: the Geospatial Reasoning Segmentation dataset (GRES), comprising 27,615 pixel-level annotations paired with natural language reasoning-segmentation queries across 9,205 images, and PreGRES, a fine-tuning dataset aggregated from existing remote-sensing datasets consisting of over 1 million question-answer pairs. These datasets enable LISAT to handle scale, resolution, and complexity, in aligning textual queries with remote-sensing and top-down imagery. LISAT achieves significant performance gains over state-of-the-art geospatial and open-domain models. Specifically, LISAT outperforms existing geospatial foundation models, such as RS-GPT4V, by over 10.04% on BLEU-4 on remote-sensing visual description tasks and outperforms state-of-the-art open-domain models on remote-sensing reasoning segmentation by 143.36% on gIoU.

## 6.2 Related Work

Semantic segmentation has been an essential task in remote-sensing with applications ranging from urban planning (135; 136; 137), economic assessment (138), precision agriculture (125), resource management (135; 139), and environmental protection (126). A key challenge facing such models, however, is that they are typically constrained to rigid, task-specific models that fail to generalize across applications without significant fine-tuning and adjustment despite using identical imagery and label ontologies. Recently, however, since the emergence of vision language models (VLMs) as a predominant paradigm (140; 141; 142), there has been a renewed interest in multimodal foundation models that are capable of responding to/answering *arbitrary* natural language (or multimodal) queries. Models such as GPT-4 (143) and LLaVA (141; 142) have expanded on this by allowing users to provide an image along with a natural language query, to solve tasks such as visual description and visual question answering.

### Remote-Sensing Datasets for Multimodal Learning

Semantic segmentation in remote sensing has long been constrained by a lack of large-scale datasets that combine fine-grained spatial annotations with multimodal supervision. Well-established benchmarks such as DeepGlobe (144; 145; 146) have advanced geospatial vision tasks by providing imagery annotated for classification, detection, and basic segmentation.

However, these datasets do not support the kind of complex, query-driven interaction that modern vision-language models require. To support multimodal tasks, several remote sensing datasets have emerged at the image or region level. Datasets like **Sydney-Captions** (147), **RSICD** (148), **NWPU-Captions** (147), **RSITMD** (149), and **UCM-Captions** (147) enable captioning and image-text retrieval. While useful for high-level semantic understanding, they are individually small in scale, and combining them will help improve text generation. More recent efforts, such as **VRSBench** (150) and **GeoChatInstruct** (132) have expanded multimodal learning to region-level tasks like grounded image captioning, region-specific question answering, and visual grounding. These datasets are built on existing remote sensing datasets (e.g., (151; 152)) and use rule-based or GPT-based methods to automatically generate textual descriptions for objects or regions within images. While they support region-level reasoning through bounding box annotations, they do not include pixel-level ground truth, which is necessary for supervised training in segmentation tasks. Datasets that do provide segmentation supervision, such as **FloodPrompt** (139) and **RefSegRS** (153), are often domain-specific or limited in scale, with **RefSegRS** offering only 4,420 images. The datasets we introduce, **PreGRES** and **GRES**, are specifically developed to address these limitations. Together, they provide a unified pipeline of detailed spatial annotations paired with natural language, enabling the training of models that can both understand and segment remote sensing imagery.

## Geospatial Foundation Models

Recent geospatial foundation models have adapted the foundation model paradigm to remote sensing, enabling multi-task capabilities for tasks like captioning, visual question answering, and object detection. **EarthGPT** (130) unifies a wide range of multi-sensor RS tasks, including scene classification, image captioning, and object detection, using a large-scale multimodal dataset derived from several task-specific datasets (see section 6.3). **TEOChat** (131) introduces temporal reasoning for applications such as change detection and damage assessment, demonstrating strong performance on temporal sequence tasks, but struggles on more general descriptions. **GeoChat** (132) supports region-specific dialogue and visual grounding, enabling fine-grained interaction with high-resolution RS imagery, while **SkyEyeGPT** (154) achieves notable performance on image-level and region-level vision-language tasks with a streamlined instruction-following architecture. **RS-GPT4V** (129) emphasizes fine-grained object understanding and complex scene reasoning, leveraging a hierarchical instruction-following approach. While these models represent significant progress across a wide range of tasks, such models have been limited by their ability to produce only *natural language outputs*. Our proposed work, **LISAT** addresses this limitation by natively producing segmentation masks in addition to answering natural language queries.

## Reasoning Segmentation

Beyond just producing segmentation masks for single classes, a goal of LISAT is to perform “reasoning segmentation,” the task of generating a segmentation mask from a complex or implicit query text (Figure 6.1). Two overarching approaches have been developed for this task. LISA (127) introduced this concept with an embedding-as-mask approach, allowing segmentation via a [SEG] token which is decoded into a segmentation mask using a SAM decoder (155). PixelLM (133) expanded on this method by leveraging a lightweight pixel decoder and segmentation codebook to improve multi-target differentiation in the same paradigm, while GLaMM (156) also targeted the granularity problem through additional focused data. GSVA (157) extended the [SEG] paradigm by introducing a [REJ] token to handle ambiguous or absent targets in queries. In the second paradigm, models such as Shikra (158), Kosmos-2 (159) and others (141; 142; 160; 161) focus on solving reasoning segmentation tasks with natural language alignment: represent visual concepts as sequences of natural language tokens (such as the literal coordinates of a bounding box). Despite these advances, existing models often fall short when applied to remote sensing due to challenges like varying spatial resolutions, fine differences between target classes, and the lack of domain-tailored datasets (See Table 6.2). Our proposed work, LISAT, extends the embedding-as-mask approach to top-down remote-sensing data.

## 6.3 Geospatial Reasoning Segmentation Dataset

The development of vision-language models (VLMs) for remote sensing has been hindered by the lack of high-quality remote sensing imagery paired with natural language data, a key challenge outlined in our introduction. Unlike natural image datasets, remote-sensing data require fine-grained, context-aware segmentation that accounts for extreme variations in scale, subtle object differences, and the ability to reason across complex spatial relationships. To help alleviate this need, we introduce the Geospatial Reasoning Segmentation Dataset (GRES), a collection of vision and language data designed around remote-sensing applications. **GRES** consists of two core components: **PreGRES**, a dataset consisting of over 1M remote-sensing specific visual instruction-tuning Q/A pairs for pre-training geospatial models, and **GRES**, a semisynthetic dataset specialized for reasoning segmentation of remote-sensing data. With this structure of **GRES**, we enable LISAT to overcome both data scarcity and the domain transfer limitations faced by general-purpose models. The dataset is specifically designed to handle scale variability, object diversity, and complex reasoning queries, making it a critical resource for advancing geospatial VLMs.

### PreGRES

**PreGRES** is a large-scale structured collection of existing smaller-scale geospatial datasets designed for fine-tuning vision-language models in remote sensing applications. It integrates multiple sources, each contributing to different aspects of geospatial data understanding. The

datasets within GRES provide coverage across image captioning, visual question answering, and visual grounding tasks:

1. **Image Captioning:** NWPU-Captions (162), RSICD (148), RSITMD (149), Sydney-Captions (147), and UCM-Captions (147). Each contributes paired image-text data, and contains long-form descriptions of top-down imagery across different geospatial environments, increasing the diversity of language supervision.
2. **Visual Question Answering (VQA):** RSVQA\_LR (163), RSVQA\_HR (163), FloodNet (164), and RSIVQA (165). Each of these datasets consists of structured question-answer pairs and supports reasoning over aerial and satellite images, covering tasks such as object identification, scene understanding, and disaster assessment.
3. **Visual Grounding / Region-Level Captioning:** DIOR-RSVG (151) provides paired text-image data for object localization and spatial reference resolution, and NWPU-RESISC45 (166) supplies scene classification labels.

Overall, PreGRES consists of 119,279 images and 1,204,993 question-answer pairs and is used in the first-stage pre-training of the LISAT model, enabling general-purpose geospatial question-answering in the final LISAT model. For more details on dataset composition, see Table D.5.

## GRES

GRES is a semi-synthetic dataset designed explicitly for geospatial reasoning segmentation. Each sample in GRES consists of an image, a natural language query referring to a single object in that image, and a pixel-level segmentation mask (See Figure 6.2 for an example of a GRES query/image pair). This task allows us to train the LISAT model to correctly localize images at a pixel level within the scene, even in the case of multiple objects requiring disambiguation.

To build the dataset, we begin with a subset of the xView dataset (146) consisting of 26,541 high-resolution satellite images spanning approximately 1,400 square kilometers, covering more than 60 classes. xView consists of paired images and object detections within the images in bounding box form. To convert xView images/annotations to GRES annotations/images, we follow the process overviewed in Figure 6.2.

Given an input image of size  $512 \times 512$ , we divide it into 4 quadrants, where the top-left quadrant is defined by  $0 \leq x \leq 255, 0 \leq y \leq 255$ ; the top-right quadrant is defined by  $256 \leq x \leq 511, 0 \leq y \leq 255$ ; the bottom-left quadrant is defined by  $0 \leq x \leq 255, 256 \leq y \leq 511$ ; the bottom-right quadrant is defined by  $256 \leq x \leq 511, 256 \leq y \leq 511$ .

In the first part of the pipeline, we need to generate a “disambiguating query” that selects for a single object within the scene from the large set of objects. To do so, we first filter the scenes for two key objectives: (1) uniqueness (i.e., can objects be easily disambiguated with a natural language query), and (2) interest (i.e., are the objects visually interesting)

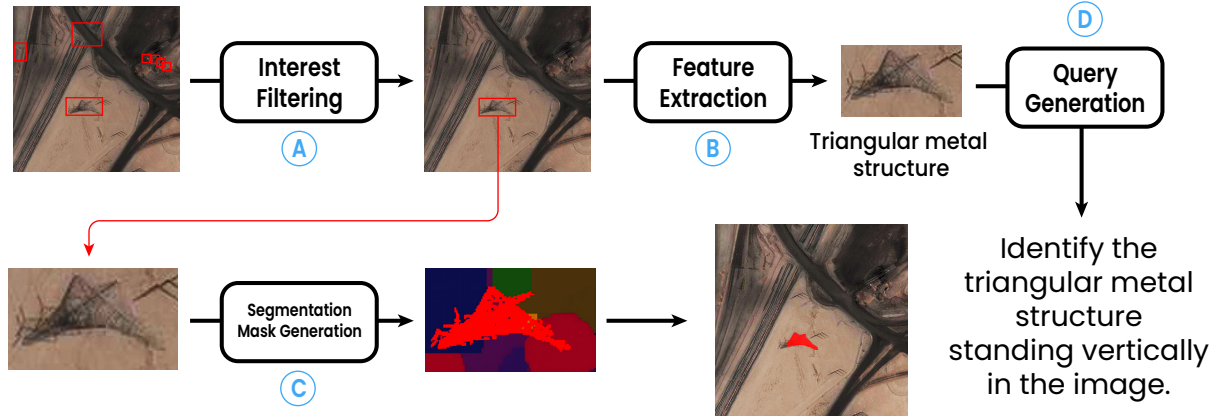


Figure 6.2: To generate synthetic data, we start with a seed detection dataset (**xView**). We then filter detections for those that are both visually interesting and highly distinguishable (A). For that detection, we then generate a natural language description (B) and a pixel-wise segmentation mask (C). Finally, the natural language description is used to generate a localization query (D). Together, the segmentation mask and the query form a ground-truth pair for the LISAT reasoning segmentation fine-tuning.

(Figure 6.2, A). An object is considered “unique” in an image if it is one of less than 2 detections of its class in its respective quadrant, and an object is considered “visually interesting” if it belongs to a class appearing in less than 50% of the overall subset of **xView** detections. Comprehensive statistics of object categories after filtering are available in Table D.1. To ensure a balanced evaluation, our dataset includes queries with and without explicit spatial references, each with a 50% probability.

After the filtering stage, we convert the object detection to a query using a set of structured queries to a large vision and language model trained on natural images (in our case, GPT-4v (143), Figure 6.2, B). In the first prompting stage, we ask the VLM to identify unique characteristics of the class within the bounding box by asking the model to “Find visual features (color, shape, size, etc.) that help find or segment {class\_name} in the image.”. We then ask the VLM to come up with a sentence describing the object in the bounding box within the scene using the collected unique characteristics (See the full prompt in section D.2). Given these features, we prompt the VLM again with the full image, along with other detections in the image and the position of the bounding box to produce a query (see the full prompt in section D.2, Figure 6.2, D).

In the second part of the pipeline (Figure 6.2, C), we need to generate the pixel-based mask from the bounding box. To do this, we leverage a GeoSAM model (167) with a custom high-resolution inference configuration (128 points per side, 0.95 prediction IoU threshold, and 0.95 stability score with an 80-pixel minimum mask region area) to produce a part-wise segmentation of each bounding box. We then add any sub-parts that cover more than 80px of the underlying bounding box to the final pixel mask.

We then asked the VLM to rephrase each query in two separate ways, which added to the initially generated query, giving us 3 queries per image. This pipeline’s overall results in a dataset consisting of 9,205 images and 27,615 natural language queries/answers within those images. From this dataset, we generate train, test, and validation splits consisting of 7,205, 1,500, and 500 images, respectively.

## 6.4 Training LISAt for Geospatial Reasoning Segmentation

Inspired by LISA (127), LISAT integrates a multimodal large language model (LLM) with a segmentation model. The multimodal LLM processes both textual and visual inputs, leveraging datasets that contain image-text pairs for instruction-following and reasoning (142) while the segmentation model uses a dataset designed for high-quality mask generation (155). An overview of the architecture is given in Figure 6.3.

### Geospatial Multimodal Language Models

While LISA (127) leverages a pre-trained LLaVA (141; 142) model as a vision and language backbone, we found that leveraging LLaVA alone was insufficient to capture the range of queries and visual variance in remote-sensing applications. To solve this problem, in the first stage of our training process, we trained a remote-sensing-specific multimodal large language model to serve as the base MLLM for the segmentation backbone. Our architecture generally follows LLaVA (141; 142) with several modifications for remote-sensing applications.

For the base language model, we leverage default Vicuna-7B (168) to embed a text query  $\mathbf{X}_l$ . For the visual backbone, LISAT adopts the Remote-CLIP ViT-L/14 encoder (169) to extract visual features from an input image  $\mathbf{X}_v$ . To align visual representations with the language model’s word embedding space, we use a simple linear projection matrix to produce a sequence of visual tokens that match the dimensionality of the word embeddings in the language model. A pre-trained Vicuna base model combined with the vision encoder is further pre-trained on PreGRES (see section 6.5) with LoRA (170) prior to being trained on GRES. We refer to this pre-trained variant as LISAT<sub>PRE</sub>.

### Preliminaries

Existing multimodal LLMs for remote sensing, such as RS-GPT4V (129) and EarthGPT (130), support images and text as input but output only text. To produce segmentation masks, LISAT leverages the “embedding-as-a-mask” paradigm introduced by LISA (127), and expands the LLM vocabulary with a new token,  $\langle \text{SEG} \rangle$ , which represents segmentation requests. When the model produces an output containing the  $\langle \text{SEG} \rangle$  token, we extract the final layer embedding of that token, and project it via an MLP layer to the query space of



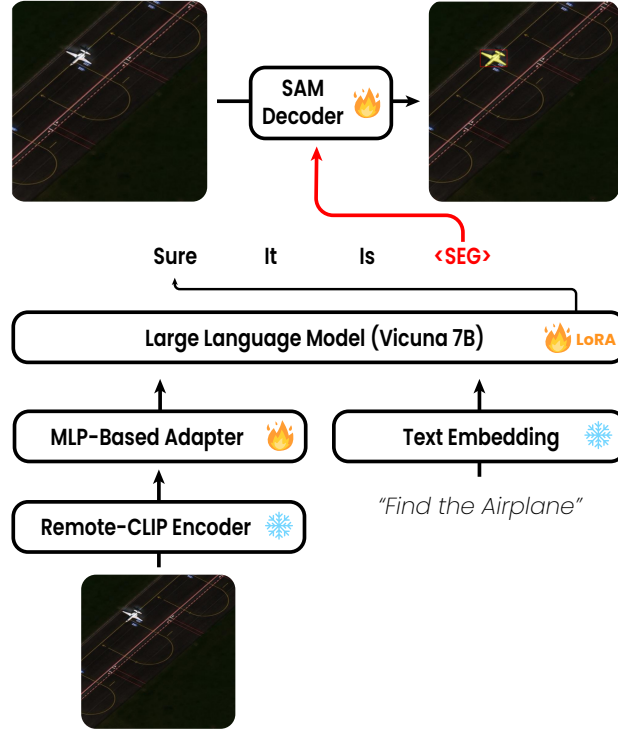


Figure 6.3: LISAT integrates a geospatial multimodal large language model (MLLM) with a segmentation decoder to enable reasoning-based segmentation. LISAT first pre-trains a Remote-CLIP-based MLLM on PreGRES before fine-tuning on GRES. We then expand the LMM vocabulary with a segmentation token (**<SEG>**), whose final-layer embedding is projected into the SAM segmentation query space and combined with image features to produce a segmentation mask.

a SAM-based segmentation decoder (155). The segmentation decoder combines the query-projected final embedding and a set of visual features extracted from the base image to produce a final segmentation mask  $\hat{\mathbf{M}}$ .

## Training Objectives

LISAT is trained end-to-end with a loss function that combines text generation and segmentation objectives. The total loss  $\mathcal{L}$  is the weighted sum of two components:

$$\mathcal{L} = \lambda_{txt} \mathcal{L}_{txt} + \lambda_{mask} \mathcal{L}_{mask}. \quad (6.1)$$

where the text generation loss  $\mathcal{L}_{txt}$  is an autoregressive cross-entropy loss:

$$\mathcal{L}_{txt} = \text{CE}(\hat{\mathbf{y}}_{txt}, \mathbf{y}_{txt}). \quad (6.2)$$

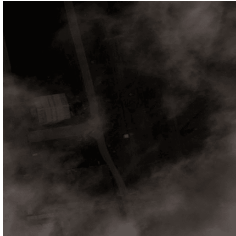
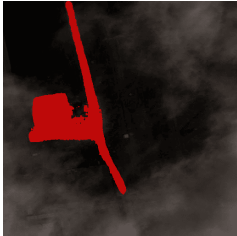
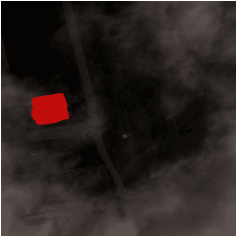
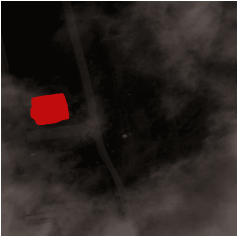








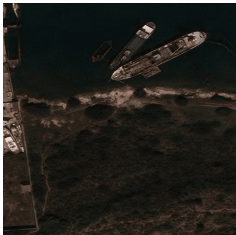
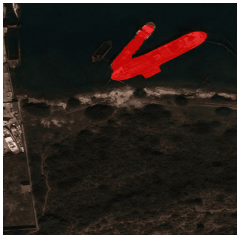
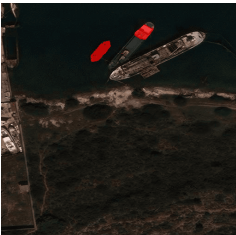
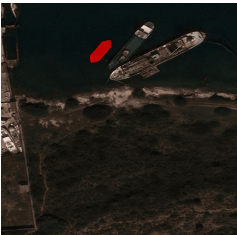
Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Locate the building with a large rectangular structure, dark roof, and symmetrical window patterns.				
Identify the facility in the center-left of the image.				
Identify the damaged building in the center of the image.				
<b>Failure Case:</b> Locate the dark, elongated rectangular shape with a red outline against the dark background to identify the barge.				

Table 6.1: Qualitative examples of the segmentations generated by LISAT on the GRES dataset.

and the segmentation loss  $\mathcal{L}_{mask}$  consists of a per-pixel binary cross-entropy (BCE) loss and a DICE loss, weighted by  $\lambda_{bce}$  and  $\lambda_{dice}$ :

$$\mathcal{L}_{mask} = \lambda_{bce} \text{BCE}(\hat{\mathbf{M}}, \mathbf{M}) + \lambda_{dice} \text{DICE}(\hat{\mathbf{M}}, \mathbf{M}). \quad (6.3)$$

## 6.5 Experimental Results

**Implementation Details:** LISAT and LISAT<sub>PRE</sub> are trained on eight DGX A100 80GB GPUs. In the first stage, we pretrain LISAT<sub>PRE</sub> (context length = 2048) using LoRA (170)

for 1 epoch on PreGRES (described in section 6.3) with next-token prediction cross-entropy loss. We employ the AdamW optimizer (171) with a learning rate of  $3e^{-4}$  and a cosine-decay learning rate scheduler, setting the batch size to 2 and gradient accumulation steps to 6.

In the second stage, we train LISAT using GRES, as well as two traditional natural image referring segmentation datasets, FP-Ref-COCO (172) and ReasonSeg (127). LoRA is applied to LISAT<sub>PRE</sub>, while the SAM decoder undergoes full fine-tuning. The learning rate is set to  $3e^{-4}$ , with all other configurations remaining the same. For the loss function, we empirically found that setting the weight for text generation loss ( $\lambda_{txt}$ ) and mask loss ( $\lambda_{mask}$ ) to 1.0, while the binary cross-entropy loss (BCE) ( $\lambda_{bce}$ ) and Dice loss ( $\lambda_{dice}$ ) are assigned weights of 2.0 and 0.5, respectively performs better. The total training time was approximately 12 hours on eight DGX A100 80GB GPUs.

**Evaluation Protocol:** We use the GRES test set to evaluate segmentation performance. We focus on two subsets of the GRES test set, Small and Large, to evaluate performance on small and large objects, respectively. We define a threshold of 500 pixels<sup>2</sup> and categorize any object in the test set that covers an area less than the threshold to be **Small** and bigger to be **Large**. We evaluate segmentation performance using generalized Intersection-over-Union (gIoU) and cumulative Intersection-over-Union (cIoU) (127). To evaluate the performance of our approach on traditional vision and language tasks, we use several existing datasets, including NWPU-Captions (162), UCM-Captions (147), Sydney-Captions (147), and RSICD (148). Following prior work, we report standard evaluation metrics: BLEU (173), CIDEr (174), and SPICE (175).

## Segmentation

Table 6.2 compares LISAT with LISA-7B-v1 and LISA-13B-Llama2-v1 (127) across different dataset configurations (All, Small, Large). LISAT consistently and significantly outperforms both natural-image trained referring segmentation models. In particular, for smaller objects, LISAT has larger relative gains compared to large models, suggesting that LISAT is more effective for capturing fine-grained spatial details, which is important for applications involving dense scenes or small-scale features in remote sensing imagery.

Some qualitative examples are given in Table 6.1. The first three rows represent success cases, where LISAT correctly identifies and localizes objects based on the queries. In the first, LISAT correctly segments the building against a noisy background, and when many of the ground features match the visual features of the target object. In the second and third, LISAT correctly identifies the key object of interest, ignoring other potential distractor objects. In the failure case, LISAT fails to correctly identify the barge alone from the two ships, likely due to the color patterns on the first ship, but still manages to outperform LISA, which only focuses on the larger ship objects.

Figure 6.4 demonstrates the influence of training dataset size on LISAT’s performance. With an increasing number of training images, LISAT demonstrates notable improvements in both cIoU and gIoU scores. These results indicate that LISAT benefits from larger training

Model	Obj. Size	cIoU	gIoU
LISA-7B	All	0.122 $\pm$ 0.014	0.113 $\pm$ 0.007
	Small	0.104 $\pm$ 0.022	0.062 $\pm$ 0.008
	Large	0.157 $\pm$ 0.017	0.222 $\pm$ 0.013
LISA-13B (llama2)	All	0.122 $\pm$ 0.014	0.139 $\pm$ 0.006
	Small	0.106 $\pm$ 0.016	0.089 $\pm$ 0.007
	Large	0.148 $\pm$ 0.018	0.244 $\pm$ 0.019
PixelLM-7B	All	0.101 $\pm$ 0.011	0.142 $\pm$ 0.006
	Small	0.069 $\pm$ 0.009	0.094 $\pm$ 0.006
	Large	0.142 $\pm$ 0.019	0.243 $\pm$ 0.014
PixelLM-13B	All	0.145 $\pm$ 0.013	0.162 $\pm$ 0.008
	Small	0.102 $\pm$ 0.015	0.106 $\pm$ 0.008
	Large	0.204 $\pm$ 0.028	0.277 $\pm$ 0.014
LISAT (Ours)	All	<b>0.245</b> $\pm$ 0.023	<b>0.275</b> $\pm$ 0.009
	Small	<b>0.232</b> $\pm$ 0.024	<b>0.240</b> $\pm$ 0.009
	Large	<b>0.250</b> $\pm$ 0.029	<b>0.348</b> $\pm$ 0.015

Table 6.2: Performance of LISAT against LISA-7B-v1, LISA-13B-Llama2-v1, PixelLM-7B and PixelLM-13B on GRES across different object sizes. LISAT-7B consistently outperforms the baseline models, particularly in the Small object category.

Segmentation Model	cIoU	gIoU
GeoSAM (167)	0.220 $\pm$ 0.019	0.238 $\pm$ 0.007
SAM (155)	<b>0.245</b> $\pm$ 0.023	<b>0.275</b> $\pm$ 0.009

Table 6.3: Comparison of LISAT’s performance using GeoSAM vs. SAM for segmentation on the A11 dataset configuration.

datasets, thereby exhibiting some good scaling properties, as its segmentation performance improves with more data, particularly for small objects.

Table 6.3 compares LISAT’s performance using GeoSAM and SAM as base segmentation models on the A11 dataset. While both models yield competitive results, SAM achieves slightly higher cIoU (0.245) and gIoU (0.275) than GeoSAM. This suggests that, despite being designed for geospatial tasks, GeoSAM alone without specific language-aligned fine-tuning, may be limited by training-specific biases, whereas SAM’s broader training on diverse natural images enables more adaptable feature extraction, leading to improved segmentation performance.

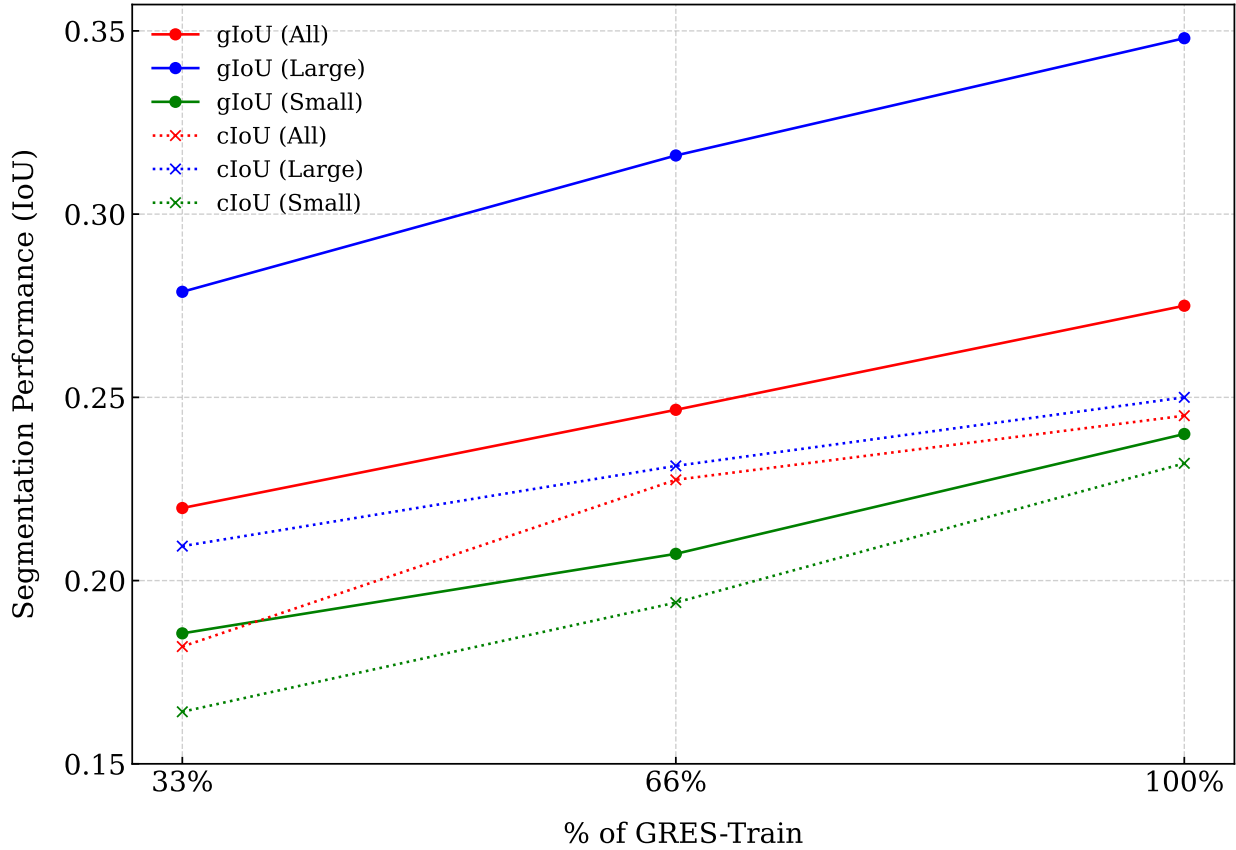


Figure 6.4: Scaling behavior of LISAT on the GRES dataset. While adding additional data is helpful, even with 7K *training* images (the full GRES dataset), we observe the beginning of a plateau in performance, particularly on cIoU scores. This suggests that more data alone may not be helpful, and instead, we may need additional data variance outside the **xView** classes.

## Captioning and Question-Answering

On the UCM-Captions dataset (Table 6.4), LISAT<sub>PRE</sub> achieves the highest BLEU-4 (72.34) and CIDEr (355.32) scores, surpassing previous geospatial models such as RS-GPT4V (129) and post-processing methods (179), as well as general-purpose vision-language models such as LLaVA-v1.5 and LLaVA-v1.6 (141; 142). For NWPU-Captions (Table 6.5), LISAT<sub>PRE</sub> achieves the highest BLEU-4 score and matches the best SPICE performance, outperforming prior geospatial captioning models such as MLCA-Net (162) and multimodal attention-based methods (148). General-purpose vision-language models (LLaVA-v1.5 and LLaVA-v1.6) (141; 142) perform significantly worse, highlighting the benefits of domain-specific training. Similar trends are observed on RSICD (Table D.6) and Sydney-Captions (Table D.7).

Table 6.6 presents the performance of LISAT<sub>PRE</sub> on the RSVQA-LR dataset across Count,

Method	BLEU-4	CIDEr
SAA (163)	64.77	294.51
SD-RSIC (176)	53.80	213.20
RTRMN (semantic) (177)	35.87	180.25
RTRMN (statistical) (177)	63.93	312.70
SVM-D BOW (178)	51.95	271.42
SVM-D CONC (178)	59.42	292.28
Post-processing (179)	62.62	309.64
LLaVA-v1.5-7b (142)	5.54	32.67
LLaVA-v1.6-7b (141)	5.44	23.86
RS-GPT4V (129)	65.74	333.23
LISA-7B (baseline)	0.00	0.00
LISA-7B (fine-tuned on GRES)	8.73	59.96
LISAT <sub>PRE</sub> (Ours)	<b>72.34</b>	<b>355.32</b>

Table 6.4: Comparison of captioning performance on the UCM-Captions dataset. Results are reported for BLEU-4 and CIDEr metrics.

Method	BLEU-4	SPICE
CSMLF (106)	47.1	26.5
Multimodal (147)	45.5	27.6
Attention (hard) (148)	46.4	28.4
FC-Att (180)	46.9	28.3
MLCA-Net (162)	47.8	28.5
LLaVA-v1.5-7b (142)	4.8	11.1
LLaVA-v1.6-7b (141)	2.9	8.7
EarthGPT(130)	65.5	<b>32.2</b>
LISA-7B (baseline)	0.00	0.00
LISA-7B (fine-tuned on GRES)	39.9	19.52
LISAT <sub>PRE</sub> (Ours)	<b>65.8</b>	<b>32.2</b>

Table 6.5: Comparison of captioning performance on the NWPU-Captions dataset. Results are reported for BLEU-4 and SPICE metrics.

Presence, and Comparison categories. The model achieves the highest Presence accuracy (92.36) and Comparison accuracy (92.20), indicating strong performance in these tasks. In contrast, models such as LLaVA-1.5 and InternLM-XC2 report lower scores in Count and Presence. These results suggest that LISAT<sub>PRE</sub> effectively handles multimodal reasoning and task-specific fine-tuning, particularly in Presence-based evaluations.

The ablation study in Table 6.7 evaluates different vision encoders and language models

Model	Count	Presence	Comparison
RSVQA (163)	67.01	87.46	81.50
EasyToHard (181)	69.22	90.66	87.49
Bi-Modal (182)	<b>72.22</b>	91.06	91.16
SHRNet (183)	73.87	91.03	90.48
LLaVA-1.5 (142)	26.81	54.72	66.22
InternLM-XC2 (184)	26.91	55.74	64.89
RS-GPT4V (129)	-	91.17	91.70
GeoChat (185)	-	91.09	90.33
Full-FT (129)	70.48	91.10	92.23
RS-GPT4V-LoRA-FT (129)	70.34	92.24	92.10
RS-GPT4V-MoE-LoRA-FT (129)	71.06	91.10	<b>92.55</b>
LLaVA-v1.5-7b (142)	18.66	53.98	66.22
LLaVA-v1.6-7b (141)	19.65	57.53	62.32
LISA-7B (baseline)	0.00	0.00	0.00
LISA-7B (fine-tuned on GRES)	25.86	79.80	84.41
LISAT <sub>PRE</sub> (Ours)	70.24	<b>92.36</b>	92.20

Table 6.6: Performance on RSVQA-LR (% accuracy).

for LISAT<sub>PRE</sub> on the UCM-Captions dataset. Among the vision encoders, RemoteCLIP (which we use in LISAT) significantly outperforms both Geo-CLIP and Sat-CLIP on all domains, while slightly outperforming the base CLIP models. Models using LLaMA 2 as a base LLM are worse than Vicuna. These findings highlight that both the vision encoder and the language model play crucial roles, with RemoteCLIP and Vicuna forming the most effective pairing for remote sensing imagery.

## Limitations and Failure Cases

While LISAT outperforms all existing reasoning segmentation models, it is not perfect. section D.4 highlights examples of failure cases in our pipeline. In some instances, LISAT struggles to produce accurate predictions when images are cloudy or when key features are obscured. Other challenges arise when the query is too vague like ‘‘Identify the plane in the bottom-right of the image.’’ while there are several planes in the bottom right corner of the image. It is also the case when similar objects to the target item appear in the image, leading to ambiguity. We hypothesize that training on a larger dataset and refining the query design could help mitigate these issues. Another issue arises from the ground truth masks generated by GeoSAM in the GRES dataset. In some cases, the underlying ground truth mask is incorrect, and LISAT is occasionally penalized even when making correct predictions, as demonstrated in section D.4.

Vision Encoder	LLM	BLEU-4	CIDEr	SPICE
CLIP	Llama 2	69.03	328.82	52.21
CLIP336	Llama 2	66.97	324.61	50.46
SAT-CLIP	Llama 2	8.82	30.41	8.15
Geo-CLIP	Llama 2	12.77	44.64	11.67
RemoteCLIP	Llama 2	68.31	330.94	52.17
CLIP	Vicuna	66.68	329.32	52.00
CLIP336	Vicuna	68.28	324.89	51.55
SAT-CLIP	Vicuna	16.87	63.92	15.08
Geo-CLIP	Vicuna	24.56	109.20	21.15
RemoteCLIP	Vicuna	<b>72.34</b>	<b>355.32</b>	<b>54.15</b>

Table 6.7: Ablations of the base language model and visual feature extractor for LISAT<sub>PRE</sub> on the UCM-Captions dataset.

## 6.6 Discussion

In this chapter, we introduce LISAT, an open-source, open-data foundation model for geospatial reasoning segmentation, and GRES, an open dataset to help support applications in remote-sensing referring segmentation. LISAT is only the first step towards models that can produce text and task-specific outputs such as masks and boxes when reasoning about a geospatial world. Future work can focus on scaling LISAT for use with large rasters, integrate LISAT with additional segmentation models other than SAM and GeoSAM, or incorporate multimodal/hyperspectral data sources. Overall, we hope that LISAT lays the groundwork for future advancements in geospatial artificial intelligence, paving the way for more sophisticated models that integrate vision and language to better understand and interact with our dynamic geospatial world.

## Impact Statement

This chapter presents advancements in reasoning segmentation for remote sensing tasks. LISAT is a method that is able to reason over arbitrary remote sensing images and output both explanations and segmentation masks for objects of interest. These kinds of workflows are extremely common across multiple fields. For example, disaster management personnel may want to know which roads leading to an airport are undamaged, and why. LISAT is the first such model that can simultaneously answer both components of such questions.

Broadly, LISAT has impacts in numerous domains such as environmental monitoring, urban planning, and search and rescue. However, one of the biggest uses of satellite imaging is surveillance. Being cognizant of this, our work is primarily based on datasets that have



been widely adopted by the remote sensing community over interesting, cluttered scenes that do not capture any individual entity.

We encourage responsible deployment and continued discourse on the implications of geospatial AI in real-world applications.

# Chapter 7

## Conclusion

This thesis explored methods to advance computer vision and multimodal systems across diverse domains, including precision segmentation in materials science, latency-optimized small object detection in ultra-high-resolution imagery, and advanced multimodal learning for robotics and geospatial analysis. Despite the varied applications, a unifying theme emerged: the efficient development of scalable, generalizable models capable of processing complex visual and contextual information.

In chapter 2, we addressed the challenge of achieving high precision in semantic segmentation of 3D volumes composed of ultra-high-resolution image slices, with a focus on critical applications such as lithium metal battery defect detection. By combining a transformer-based encoder with a CNN-based decoder, we demonstrated how this hybrid architecture can significantly improve the detection of dendrite growth in lithium metal batteries, contributing to a better understanding of dendrite morphology and its implications for battery design.

chapter 3 focused on reducing latency in large-scale visual detection through segmentation tasks, with an emphasis on small barcode detection in ultra-high-resolution images. By employing a two-stage pipeline that combines a modified region proposal network with precise semantic segmentation, we achieved substantial improvements in inference speed while maintaining high accuracy. This work highlighted how architectural design and processing strategies directly impact the real-time performance of computer vision systems.

chapter 4 shifted focus toward multimodal learning, particularly in the context of active perception. In this chapter, we introduced RPTx, a sensorimotor pretraining model for robotics that applies a non-auto-regressive BERT-style masked token prediction approach inspired by RPT (3). While RPT performed well on in-domain data, RPTx struggled to scale effectively when applied to the broader, more diverse OXE dataset. This limitation highlighted the challenges of adapting BERT-style masked prediction methods to large-scale multimodal robotic data, especially when such data do not benefit from the sequential modeling strengths leveraged in auto-regressive large language models.

chapter 5 addressed this gap with LLARVA, a vision-action instruction-tuned model employing a next-token prediction strategy similar to auto-regressive large language models.

By aligning vision, action, and language through structured instruction tuning, LLARVA achieved superior scalability and generalization on the same OXE dataset, where RPTx fell short. However, LLARVA’s higher computational latency revealed a key tradeoff between model performance and deployment efficiency, pointing to future opportunities for optimization.

Finally, chapter 6 extended the investigation of auto-regressive multimodal models into the geospatial domain with LISAT, a language-instructed segmentation assistant for satellite imagery. LISAT achieved strong benchmark performance, surpassing existing models on reasoning segmentation tasks. This work validated that combining carefully chosen vision and language encoders can significantly enhance task-specific performance. Furthermore, the release of the PreGRES and GRES datasets, along with LISAT and LISAT<sub>PRE</sub> open-source models, established a foundation for future research in geospatial AI, demonstrating the importance of both high-quality pre-training and fine-tuning data as well as adaptable model architectures.

Across all chapters, this thesis highlights that auto-regressive multimodal models that utilize language can achieve robust, scalable performance when training objectives, model architecture, and data are carefully aligned with task requirements. In robotics, shifting from masked token prediction to next-token prediction improved performance and generalization. In geospatial reasoning segmentation, thoughtful encoder pairing and dataset design were essential to success. For precision tasks, architectural refinement combined with latency-aware design proved to be necessary.

Looking forward, key opportunities lie in improving the computational efficiency of large multimodal models, exploring their combination with reinforcement learning for active perception, and extending pre-training paradigms across broader domains. Open data sharing and community-driven model development will also be useful in driving progress.

In summary, this work contributes not only technical advancements in model design and training strategies but also practical insights into how multimodal systems can be better adapted for real-world challenges in materials science, small object detection, robotics, and geospatial intelligence.

# Bibliography

- [1] Y. Huang, D. Perlmutter, A. Fei-Huei Su, J. Quenum, P. Shevchenko, D. Y. Parkinson, I. V. Zenyuk, and D. Ushizima, “Detecting lithium plating dynamics in a solid-state battery with operando x-ray computed tomography using machine learning,” *npj Computational Materials*, vol. 9, no. 1, p. 93, 2023.
- [2] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.
- [3] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik, “Robot learning with sensorimotor pre-training,” in *Conference on Robot Learning*, pp. 683–693, PMLR, 2023.
- [4] D. Niu, Y. Sharma, G. Biamby, J. Quenum, Y. Bai, B. Shi, T. Darrell, and R. Herzig, “Llarva: Vision-action instruction tuning enhances robot learning,” *arXiv preprint arXiv:2406.11815*, 2024.
- [5] Y. Liu, Q. Liu, L. Xin, Y. Liu, F. Yang, E. A. Stach, and J. Xie, “Making li-metal electrodes rechargeable by controlling the dendrite growth direction,” *Nature Energy*, vol. 2, no. 7, pp. 1–10, 2017.
- [6] O. X.-E. Collaboration, A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frueh, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake,

- J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitran, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Bajjal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin, "Open X-Embodiment: Robotic learning datasets and RT-X models." <https://arxiv.org/abs/2310.08864>, 2023.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [10] B. Yamauchi, A. Schultz, and W. Adams, "Mobile robot exploration and map-building with continuous localization," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 4, pp. 3715–3720, IEEE, 1998.

- [11] J. Quenum, I. V. Zenyuk, and D. Ushizima, “Lithium metal battery quality control via transformer–cnn segmentation,” *Journal of Imaging*, vol. 9, no. 6, 2023.
- [12] J. Quenum, K. Wang, and A. Zakhor, “Fast, accurate barcode detection in ultra high-resolution images,” in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1019–1023, 2021.
- [13] J. Quenum, I. V. Zenyuk, and D. Ushizima, “Lithium metal battery quality control via transformer–cnn segmentation,” *Journal of Imaging*, vol. 9, no. 6, p. 111, 2023.
- [14] I. V. Zenyuk, “Bridging x-ray computed tomography and computational modeling for electrochemical energy-conversion and–storage,” *Current Opinion in Electrochemistry*, vol. 13, pp. 78–85, 2019.
- [15] V. D. Veeraraghavan, L. Frenck, J. A. Maslyn, W. S. Loo, D. Y. Parkinson, and N. P. Balsara, “Evolution of protrusions on lithium metal anodes stabilized by a solid block copolymer electrolyte studied using time-resolved x-ray tomography,” *ACS Applied Materials & Interfaces*, 2021.
- [16] K. J. Harry, X. Liao, D. Y. Parkinson, A. M. Minor, and N. P. Balsara, “Electrochemical deposition and stripping behavior of lithium metal across a rigid block copolymer electrolyte membrane,” *Journal of the Electrochemical Society*, vol. 162, no. 14, p. A2699, 2015.
- [17] A. S. Ho, P. Barai, J. A. Maslyn, L. Frenck, W. S. Loo, D. Y. Parkinson, V. Srinivasan, and N. P. Balsara, “Uncovering the relationship between diameter and height of electrodeposited lithium protrusions in a rigid electrolyte,” *ACS Applied Energy Materials*, vol. 3, no. 10, pp. 9645–9655, 2020.
- [18] J. A. Lewis, F. J. Q. Cortes, Y. Liu, J. C. Miers, A. Verma, B. S. Vishnugopi, J. Tipkens, D. Prakash, T. S. Marchese, S. Y. Han, *et al.*, “Linking void and interphase evolution to electrochemistry in solid-state batteries using operando x-ray tomography,” *Nature Materials*, vol. 20, no. 4, pp. 503–510, 2021.
- [19] A. S. Ho, D. Y. Parkinson, D. P. Finegan, S. E. Trask, A. N. Jansen, W. Tong, and N. P. Balsara, “3d detection of lithiation and lithium plating in graphite anodes during fast charging,” *ACS nano*, 2021.
- [20] A. Fioravante de Siqueira, D. M. Ushizima, and S. J. van der Walt, “A reusable neural network pipeline for unidirectional fiber segmentation,” *Scientific Data*, vol. 9, no. 1, p. 32, 2022.
- [21] G. L. B. Ramalho, D. S. Ferreira, A. G. C. Bianchi, C. M. Carneiro, F. N. S. Medeiros, and D. M. Ushizima, “Cell reconstruction under Voronoi and enclosing ellipses from 3d microscopy,” *IEEE International Symposium on Biomedical Imaging (ISBI)*, Apr 2015.

- [22] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.
- [23] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [24] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.
- [25] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, “Denseaspp for semantic segmentation in street scenes,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3684–3692, 2018.
- [26] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning,” *Image Recognition*, vol. 7, 2015.
- [28] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.
- [29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” *ICLR*, 2021.
- [30] A. Hassani, S. Walton, J. Li, S. Li, and H. Shi, “Neighborhood attention transformer,” *arXiv preprint arXiv:2204.07143*, 2022.
- [31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.
- [32] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, *et al.*, “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6881–6890, 2021.
- [33] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “Segformer: Simple and efficient design for semantic segmentation with transformers,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 12077–12090, 2021.

- [34] Y. Gao, M. Zhou, and D. N. Metaxas, “Utnet: a hybrid transformer architecture for medical image segmentation,” in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pp. 61–71, Springer, 2021.
- [35] K.-B. Park and J. Y. Lee, “Swine-net: hybrid deep learning approach to novel polyp segmentation using convolutional neural network and swin transformer,” *Journal of Computational Design and Engineering*, vol. 9, no. 2, pp. 616–632, 2022.
- [36] Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 173–190, Springer, 2020.
- [37] S. Chakraborty, G. K. Sethi, L. Frenck, A. S. Ho, I. Villaluenga, H. Wantanabe, and N. P. Balsara, “Effect of yield stress on stability of block copolymer electrolytes against lithium metal electrodes,” *ACS Applied Energy Materials*, vol. 0, no. 0, p. null, 0.
- [38] C. Monroe and J. Newman, “Dendrite growth in lithium/polymer systems: A propagation model for liquid electrolytes under galvanostatic conditions,” *Journal of The Electrochemical Society*, vol. 150, no. 10, p. A1377, 2003.
- [39] C. Monroe and J. Newman, “The effect of interfacial deformation on electrodeposition kinetics,” *Journal of The Electrochemical Society*, vol. 151, no. 6, p. A880, 2004.
- [40] S. Yu and D. J. Siegel, “Grain boundary softening: a potential mechanism for lithium metal penetration through stiff solid electrolytes,” *ACS applied materials & interfaces*, vol. 10, no. 44, pp. 38151–38158, 2018.
- [41] A. Badran, D. Parkinson, D. Ushizima, D. Marshall, and E. Maillet, “Validation of deep learning segmentation of ct images of fiber-reinforced composites,” *Journal of Composites Science*, vol. 6, no. 2, 2022.
- [42] R. Sadre, B. Sundaram, S. Majumdar, and D. Ushizima, “Validating deep learning inference during chest x-ray classification for covid-19 screening,” *Scientific Reports*, vol. 11, no. 1, p. 16075, 2021.
- [43] D. Ushizima, Y. Chen, M. Alegro, D. Ovando, R. Eser, W. Lee, K. Poon, A. Shankar, N. Kantamneni, S. Satrawada, *et al.*, “Deep learning for alzheimer’s disease: Mapping large-scale histological tau protein for neuroimaging biomarker validation,” *NeuroImage*, vol. 248, p. 118790, 2022.
- [44] K. Zhang, T.-T. Nguyen, Z. Su, and A. Demortière, “Self-supervised image quality assessment for x-ray tomographic images of li-ion battery,” *npj Computational Materials*, vol. 8, no. 1, p. 194, 2022.



- [45] Z. Su, E. Decenci re, T.-T. Nguyen, K. El-Amiry, V. De Andrade, A. A. Franco, and A. Demort re, “Artificial neural network approach for multiphase segmentation of battery electrode nano-ct images,” *npj Computational Materials*, vol. 8, no. 1, p. 30, 2022.
- [46] K. N. Wood, E. Kazyak, A. F. Chadwick, K.-H. Chen, J.-G. Zhang, K. Thornton, and N. P. Dasgupta, “Dendrites and pits: Untangling the complex behavior of lithium metal anodes through operando video microscopy,” *ACS Central Science*, vol. 2, pp. 790–801, 11 2016.
- [47] D. Cao, X. Sun, Q. Li, A. Natan, P. Xiang, and H. Zhu, “Lithium dendrite in all-solid-state batteries: Growth mechanisms, suppression strategies, and characterizations,” *Matter*, vol. 3, no. 1, pp. 57–94, 2020.
- [48] J. Zhu, Z. Zhang, S. Zhao, A. S. Westover, I. Belharouak, and P.-F. Cao, “Single-ion conducting polymer electrolytes for solid-state lithium–metal batteries: Design, performance, and challenges,” *Advanced Energy Materials*, vol. 11, no. 14, p. 2003836, 2021.
- [49] C. Huang, C. L. A. Leung, P. Leung, and P. S. Grant, “A solid-state battery cathode with a polymer composite electrolyte and low tortuosity microstructure by directional freezing and polymerization,” *Advanced Energy Materials*, vol. 11, no. 1, p. 2002387, 2021.
- [50] D. Ushizima, K. Xu, and P. Monteiro, “Materials data science for microstructural characterization of archaeological concrete,” *MRS Advancements - special issue: Materials Data Science*, pp. 1–14, 2020.
- [51] K. B. Hatzell, X. C. Chen, C. L. Cobb, N. P. Dasgupta, M. B. Dixit, L. E. Marbella, M. T. McDowell, P. P. Mukherjee, A. Verma, V. Viswanathan, A. S. Westover, and W. G. Zeier, “Challenges in lithium metal anodes for solid-state batteries,” *ACS Energy Letters*, vol. 5, no. 3, pp. 922–934, 2020.
- [52] Y. Ji, J. Li, and J. Li, “Recent development of electrolyte engineering for sodium metal batteries,” *Batteries*, vol. 8, no. 10, 2022.
- [53] E. Knudsen, P. Albertus, K. Cho, A. Weber, and A. Kojic, “Flow simulation and analysis of high-power flow batteries,” *Journal of Power Sources*, vol. 299, pp. 617–628, 2015.
- [54] Y. Gao, Y. Zhao, Y. C. Li, Q. Huang, T. E. Mallouk, and D. Wang, “Interfacial chemistry regulation via a skin-grafting strategy enables high-performance lithium-metal batteries,” *Journal of the American Chemical Society*, vol. 139, no. 43, pp. 15288–15291, 2017.

- [55] W. Zhou, S. Wang, Y. Li, S. Xin, A. Manthiram, and J. B. Goodenough, “Plating a dendrite-free lithium anode with a polymer/ceramic/polymer sandwich electrolyte,” *Journal of the American Chemical Society*, vol. 138, no. 30, pp. 9385–9388, 2016.
- [56] T. Ates, M. Keller, J. Kulisch, T. Adermann, and S. Passerini, “Development of an all-solid-state lithium battery by slurry-coating procedures using a sulfidic electrolyte,” *Energy Storage Materials*, vol. 17, pp. 204–210, 2019.
- [57] A. Z. Weber, R. L. Borup, R. M. Darling, P. K. Das, T. J. Dursch, W. Gu, D. Harvey, A. Kusoglu, S. Litster, M. M. Mench, R. Mukundan, J. P. Owejan, J. G. Pharoah, M. Secanell, and I. V. Zenyuk, “A critical review of modeling transport phenomena in polymer-electrolyte fuel cells,” *Journal of The Electrochemical Society*, vol. 161, p. F1254, sep 2014.
- [58] F. De Carlo, X. Xiao, and C. Jacobsen, “Tomopy: a framework for the analysis of synchrotron tomographic data,” in *Journal of Synchrotron Radiation*, Citeseer, 2014.
- [59] D. M. Pelt, D. Gürsoy, W. J. Palenstijn, J. Sijbers, F. De Carlo, and K. J. Batenburg, “Integration of tomopy and the astra toolbox for advanced processing and reconstruction of tomographic synchrotron data,” *Journal of synchrotron radiation*, vol. 23, no. 3, pp. 842–849, 2016.
- [60] F. De Carlo, D. Gürsoy, F. Marone, M. Rivers, D. Y. Parkinson, F. Khan, N. Schwarz, D. J. Vine, S. Vogt, S.-C. Gleber, *et al.*, “Scientific data exchange: a schema for hdf5-based storage of raw and analyzed data,” *Journal of synchrotron radiation*, vol. 21, no. 6, pp. 1224–1230, 2014.
- [61] M. Brown, R. Szeliski, and S. Winder, “Multi-image matching using multi-scale oriented patches,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 510–517, IEEE, 2005.
- [62] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.
- [63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.
- [64] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, “Tversky loss function for image segmentation using 3d fully convolutional deep networks,” 2017.
- [65] N. Abraham and N. M. Khan, “A novel focal tversky loss function with improved attention u-net for lesion segmentation,” 2018.

- [66] A. Namane and M. Arezki, “Fast real time 1d barcode detection from webcam images using the bars detection method,” in *Proceedings of the World Congress on Engineering (WCE)*, vol. 1, 2017.
- [67] L. Hock, H. Hanaizumi, and E. Ohbuchi, “Barcode readers using the camera device in mobile phones,” in *2013 International Conference on Cyberworlds*, pp. 260–265, IEEE Computer Society, 2004.
- [68] M. Katona and L. G. Nyúl, “Efficient 1d and 2d barcode detection using mathematical morphology,” in *Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 464–475, Springer Berlin Heidelberg, 2013.
- [69] G. Sörös and C. Flörkemeier, “Blur-resistant joint 1d and 2d barcode localization for smartphones,” in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM ’13, Association for Computing Machinery, 2013.
- [70] C. Creusot and A. Munawar, “Low-computation egocentric barcode detector for the blind,” in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2856–2860, 2016.
- [71] C. Creusot and A. Munawar, “Real-time barcode detection in the wild.,” *IEEE Winter Conference on Applications of Computer Vision*, p. 239–245, 2015.
- [72] O. Gallo and R. Manduchi, “Reading 1d barcodes with mobile phones using deformable templates.,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, p. 1834–1843, 2011.
- [73] A. Zamberletti, I. Gallo, M. Carullo, and E. Binaghi, “Neural image restoration for decoding 1-d barcodes using common camera phones.,” vol. 1, pp. 5–11, 01 2010.
- [74] D. K. Hansen, K. Nasrollahi, C. B. Rasmussen, and T. B. Moeslund, “Real-time barcode detection and classification using deep learning,” *IJCCI*, vol. 1, pp. 321–327, 2017.
- [75] P. D. K. He, G. Gkioxari and R. Girshick, “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.
- [76] A. Bochkovskiy, C. Wang, and H. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” 2020.
- [77] A. Zharkov and I. Zagaynov, “Universal barcode detector via semantic segmentation,” *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 837–843, 2019.
- [78] J. Quenum, K. Wang, and A. Zakhori, “Fast, accurate barcode detection in ultra high-resolution images,” in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1019–1023, IEEE, 2021.

- [79] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [80] X. J. S. Wachenfeld, S. Terlunen, *Robust 1-D Barcode Recognition on Camera Phones and Mobile Product Information Display*, pp. 53–69. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
- [81] A. Mohammed, S. Yildirim, I. Farup, M. Pedersen, and Øistein Hovde, “Y-net: A deep convolutional neural network for polyp detection,” 2018.
- [82] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, “A real-time algorithm for signal analysis with the help of the wavelet transform,” *In Wavelets: Time-Frequency Methods and Phase Space. Proceedings of the International Conference*, 1987.
- [83] W. Hao, C. Li, X. Li, L. Carin, and J. Gao, “Towards learning a generic agent for vision-and-language navigation via pre-training,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13137–13146, 2020.
- [84] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022.
- [85] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *arXiv preprint arXiv:2307.15818*, 2023.
- [86] J. Devlin, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [87] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.
- [88] A. Vaswani, “Attention is all you need,” *Advances in Neural Information Processing Systems*, 2017.

- [89] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, “Real-world robot learning with masked visual pre-training,” in *Conference on Robot Learning*, pp. 416–426, PMLR, 2023.
- [90] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” *arXiv preprint arXiv:2307.15818*, 2023.
- [91] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024.
- [92] S. D. Whitehead and D. H. Ballard, “Active perception and reinforcement learning,” in *Machine Learning Proceedings 1990*, pp. 179–188, Elsevier, 1990.
- [93] L. Bartolomei, L. Teixeira, and M. Chli, “Semantic-aware active perception for uavs using deep reinforcement learning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3101–3108, IEEE, 2021.
- [94] M. Shen and J. P. How, “Active perception in adversarial scenarios using maximum entropy deep reinforcement learning,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3384–3390, IEEE, 2019.
- [95] W. Tang and M. R. Jahanshahi, “Active perception based on deep reinforcement learning for autonomous robotic damage inspection,” *Machine Vision and Applications*, vol. 35, no. 5, pp. 1–20, 2024.
- [96] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, K. Choromanski, T. Ding, D. Driess, K. A. Dubey, C. Finn, P. R. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, S. Levine, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. S. Ryoo, G. Salazar, P. R. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. H. Vuong, A. Wahid, S. Welker, P. Wohlhart, T. Xiao, T. Yu, and B. Zitkovich, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” *ArXiv*, vol. abs/2307.15818, 2023.
- [97] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. M. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. C. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. García, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Díaz,

- O. Firat, M. Catasta, J. Wei, K. S. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, “Palm: Scaling language modeling with pathways,” *J. Mach. Learn. Res.*, vol. 24, pp. 240:1–240:113, 2022.
- [98] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “Openvla: An open-source vision-language-action model,” 2024.
- [99] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi, “Instructblip: Towards general-purpose vision-language models with instruction tuning,” 2023.
- [100] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.
- [101] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” in *NeurIPS*, 2023.
- [102] H. Liu, C. Li, Y. Li, and Y. J. Lee, “Improved baselines with visual instruction tuning,” 2023.
- [103] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. H. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. R. Florence, “Palm-e: An embodied multimodal language model,” in *International Conference on Machine Learning*, 2023.
- [104] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *Conference on Robot Learning*, pp. 785–799, PMLR, 2023.
- [105] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.
- [106] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.
- [107] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.

- [108] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” <https://github.com/facebookresearch/detectron2>, 2019.
- [109] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.
- [110] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.
- [111] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [112] E. B. Avraham, R. Herzig, K. Mangalam, A. Bar, A. Rohrbach, L. Karlinsky, T. Darrell, and A. Globerson, “Bringing image scene structure to video via frame-clip consistency of object tokens,” in *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.
- [113] X. Wang and A. Gupta, “Videos as space-time region graphs,” in *ECCV*, 2018.
- [114] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori, “Object level visual reasoning in videos,” in *ECCV*, pp. 105–121, 2018.
- [115] D. Shan, J. Geng, M. Shu, and D. Fouhey, “Understanding human hands in contact at internet scale,” in *CVPR*, 2020.
- [116] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, “Image retrieval using scene graphs,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3668–3678, 2015.
- [117] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, “Scene Graph Generation by Iterative Message Passing,” in *CVPR*, pp. 3097–3106, 2017.
- [118] A. Bar, R. Herzig, X. Wang, A. Rohrbach, G. Chechik, T. Darrell, and A. Globerson, “Compositional video synthesis with action graphs,” in *ICML*, 2021.
- [119] A. Kamath, M. Singh, Y. LeCun, I. Misra, G. Synnaeve, and N. Carion, “Mdetr - modulated detection for end-to-end multi-modal understanding,” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1760–1770, 2021.
- [120] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, “Generation and comprehension of unambiguous object descriptions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 11–20, 2016.

- [121] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, “Modeling context in referring expressions,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14*, pp. 69–85, Springer, 2016.
- [122] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia, “Lisa: Reasoning segmentation via large language model,” *ArXiv*, vol. abs/2308.00692, 2023.
- [123] T.-H. Wu, G. Biamby, D. Chan, L. Dunlap, R. Gupta, X. Wang, J. E. Gonzalez, and T. Darrell, “See say and segment: Teaching lmms to overcome false premises,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13459–13469, June 2024.
- [124] J. Quenum, W.-H. Hsieh, T.-H. Wu, R. Gupta, T. Darrell, and D. M. Chan, “Lisat: Language-instructed segmentation assistant for satellite imagery,” *arXiv preprint arXiv:2505.02829*, 2025.
- [125] M. Weiss, F. Jacob, and G. Duveiller, “Remote sensing for agricultural applications: A meta-review,” *Remote sensing of environment*, vol. 236, p. 111402, 2020.
- [126] S. Subudhi, R. N. Patro, P. K. Biswal, and F. Dell’Acqua, “A survey on superpixel segmentation as a preprocessing step in hyperspectral image analysis,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 5015–5035, 2021.
- [127] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia, “Lisa: Reasoning segmentation via large language model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9579–9589, 2024.
- [128] E. Rolf, K. Klemmer, C. Robinson, and H. Kerner, “Position: Mission critical–satellite data is a distinct modality in machine learning,” in *Forty-first International Conference on Machine Learning*, 2024.
- [129] L. Xu, L. Zhao, W. Guo, Q. Li, K. Long, K. Zou, Y. Wang, and H. Li, “Rs-gpt4v: A unified multimodal instruction-following dataset for remote sensing image understanding,” *arXiv preprint arXiv:2406.12479*, 2024.
- [130] W. Zhang, M. Cai, T. Zhang, Y. Zhuang, and X. Mao, “Earthgpt: A universal multimodal large language model for multi-sensor image comprehension in remote sensing domain,” *IEEE Transactions on Geoscience and Remote Sensing*, 2024.
- [131] J. A. Irvin, E. R. Liu, J. C. Chen, I. Dormoy, J. Kim, S. Khanna, Z. Zheng, and S. Ermon, “TeoChat: A large vision-language assistant for temporal earth observation data,” *arXiv preprint arXiv:2410.06234*, 2024.



- [132] K. Kuckreja, M. S. Danish, M. Naseer, A. Das, S. Khan, and F. S. Khan, “Geochat: Grounded large vision-language model for remote sensing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27831–27840, 2024.
- [133] Z. Ren, Z. Huang, Y. Wei, Y. Zhao, D. Fu, J. Feng, and X. Jin, “Pixellm: Pixel reasoning with large multimodal model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26374–26383, 2024.
- [134] R. Hu, M. Rohrbach, and T. Darrell, “Segmentation from natural language expressions,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 108–124, Springer, 2016.
- [135] T. Gao, W. Ao, X.-A. Wang, Y. Zhao, P. Ma, M. Xie, H. Fu, J. Ren, and Z. Gao, “Enrich distill and fuse: Generalized few-shot semantic segmentation in remote sensing leveraging foundation model’s assistance,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2771–2780, 2024.
- [136] L. H. Hansen, S. B. Jensen, M. P. Philipsen, A. Møgelmoose, L. Bodum, and T. B. Moeslund, “Opentrench3d: A photogrammetric 3d point cloud dataset for semantic segmentation of underground utilities,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7646–7655, 2024.
- [137] L. Chen, Z. Qu, Y. Zhang, J. Liu, R. Wang, and D. Zhang, “Edge enhanced gcifnet: A multiclass semantic segmentation network based on edge enhancement and multiscale attention mechanism,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024.
- [138] J. Flotzinger, P. J. Rösch, C. Benz, M. Ahmad, M. Cankaya, H. Mayer, V. Rodehorst, N. Oswald, and T. Braml, “dacl-challenge: Semantic segmentation during visual bridge inspections,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 716–725, 2024.
- [139] L. Li, “Cpseg: Finer-grained image semantic segmentation via chain-of-thought language prompting,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 513–522, 2024.
- [140] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.
- [141] H. Liu, C. Li, Y. Li, B. Li, Y. Zhang, S. Shen, and Y. J. Lee, “Llava-next: Improved reasoning, ocr, and world knowledge,” 2024.

- [142] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” *Advances in neural information processing systems*, vol. 36, 2024.
- [143] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [144] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, “Deepglobe 2018: A challenge to parse the earth through satellite images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 172–181, 2018.
- [145] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, “Spacenet: A remote sensing dataset and challenge series,” *arXiv preprint arXiv:1807.01232*, 2018.
- [146] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord, “xview: Objects in context in overhead imagery,” *arXiv preprint arXiv:1802.07856*, 2018.
- [147] B. Qu, X. Li, D. Tao, and X. Lu, “Deep semantic understanding of high resolution remote sensing image,” in *2016 International conference on computer, information and telecommunication systems (Cits)*, pp. 1–5, IEEE, 2016.
- [148] X. Lu, B. Wang, X. Zheng, and X. Li, “Exploring models and data for remote sensing image caption generation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2183–2195, 2017.
- [149] Z. Yuan, W. Zhang, K. Fu, X. Li, C. Deng, H. Wang, and X. Sun, “Exploring a fine-grained multiscale method for cross-modal remote sensing image retrieval,” *arXiv preprint arXiv:2204.09868*, 2022.
- [150] X. Li, J. Ding, and M. Elhoseiny, “Vrsbench: A versatile vision-language benchmark dataset for remote sensing image understanding,” *arXiv preprint arXiv:2406.12384*, 2024.
- [151] Y. Zhan, Z. Xiong, and Y. Yuan, “Rsvg: Exploring data and models for visual grounding on remote sensing data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–13, 2023.
- [152] D. Wang, J. Zhang, B. Du, M. Xu, L. Liu, D. Tao, and L. Zhang, “Samrs: Scaling-up remote sensing segmentation dataset with segment anything model,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 8815–8827, 2023.
- [153] Z. Yuan, L. Mou, Y. Hua, and X. X. Zhu, “Rrsi: Referring remote sensing image segmentation,” *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

- [154] Y. Zhan, Z. Xiong, and Y. Yuan, “Skyeyegpt: Unifying remote sensing vision-language tasks via instruction tuning with large language model,” *arXiv preprint arXiv:2401.09712*, 2024.
- [155] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, “Segment anything,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.
- [156] H. Rasheed, M. Maaz, S. Shaji, A. Shaker, S. Khan, H. Cholakkal, R. M. Anwer, E. Xing, M.-H. Yang, and F. S. Khan, “Glamm: Pixel grounding large multimodal model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13009–13018, 2024.
- [157] Z. Xia, D. Han, Y. Han, X. Pan, S. Song, and G. Huang, “Gsva: Generalized segmentation via multimodal large language models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3858–3869, 2024.
- [158] K. Chen, Z. Zhang, W. Zeng, R. Zhang, F. Zhu, and R. Zhao, “Shikra: Unleashing multimodal llm’s referential dialogue magic,” *arXiv preprint arXiv:2306.15195*, 2023.
- [159] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, Q. Ye, and F. Wei, “Grounding multimodal large language models to the world,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [160] Y. Zhang, Z. Ma, X. Gao, S. Shakiah, Q. Gao, and J. Chai, “Groundhog: Grounding large language models to holistic segmentation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14227–14238, 2024.
- [161] S. Xuan, Q. Guo, M. Yang, and S. Zhang, “Pink: Unveiling the power of referential comprehension for multi-modal llms,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13838–13848, 2024.
- [162] Q. Cheng, H. Huang, Y. Xu, Y. Zhou, H. Li, and Z. Wang, “Nwpu-captions dataset and mlca-net for remote sensing image captioning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–19, 2022.
- [163] S. Lobry, D. Marcos, J. Murray, and D. Tuia, “Rsvqa: Visual question answering for remote sensing data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 12, pp. 8555–8566, 2020.
- [164] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. R. Murphy, “Floodnet: A high resolution aerial imagery dataset for post flood scene understanding,” *IEEE Access*, vol. 9, pp. 89644–89654, 2021.

- [165] X. Zheng, B. Wang, X. Du, and X. Lu, “Mutual attention inception network for remote sensing visual question answering,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.
- [166] G. Cheng, J. Han, and X. Lu, “Remote sensing image scene classification: Benchmark and state of the art,” *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.
- [167] R. I. Sultana, C. Lia, H. Zhua, P. Khanduria, M. Brocanellib, and D. Zhua, “Geosam: Fine-tuning sam with multi-modal prompts for mobility infrastructure segmentation,” *arXiv preprint arXiv:2311.11319*, 2023.
- [168] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, “Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality,” March 2023.
- [169] F. Liu, D. Chen, Z. Guan, X. Zhou, J. Zhu, Q. Ye, L. Fu, and J. Zhou, “Remoteclip: A vision language foundation model for remote sensing,” *IEEE Transactions on Geoscience and Remote Sensing*, 2024.
- [170] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [171] I. Loshchilov, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.
- [172] T.-H. Wu, G. Biamby, D. Chan, L. Dunlap, R. Gupta, X. Wang, J. E. Gonzalez, and T. Darrell, “See say and segment: Teaching lmms to overcome false premises,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13459–13469, 2024.
- [173] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
- [174] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.
- [175] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “Spice: Semantic propositional image caption evaluation,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part V 14*, pp. 382–398, Springer, 2016.

- [176] G. Sumbul, S. Nayak, and B. Demir, “Sd-rsic: Summarization-driven deep remote sensing image captioning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 8, pp. 6922–6934, 2020.
- [177] B. Wang, X. Zheng, B. Qu, and X. Lu, “Retrieval topic recurrent memory network for remote sensing image captioning,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 256–270, 2020.
- [178] G. Hoxha *et al.*, “A novel svm-based decoder for remote sensing image captioning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.
- [179] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, B. Patra, *et al.*, “Language is not all you need: Aligning perception with language models,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 72096–72109, 2023.
- [180] X. Zhang, X. Wang, X. Tang, H. Zhou, and C. Li, “Description generation for remote sensing images using attribute attention mechanism,” *Remote Sensing*, vol. 11, no. 6, p. 612, 2019.
- [181] Z. Yuan, L. Mou, Q. Wang, and X. X. Zhu, “From easy to hard: Learning language-guided curriculum for visual question answering on remote sensing data,” *IEEE transactions on geoscience and remote sensing*, vol. 60, pp. 1–11, 2022.
- [182] Y. Bazi, M. M. Al Rahhal, M. L. Mekhalfi, M. A. Al Zuair, and F. Melgani, “Bi-modal transformer-based approach for visual question answering in remote sensing imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2022.
- [183] Z. Zhang, L. Jiao, L. Li, X. Liu, P. Chen, F. Liu, Y. Li, and Z. Guo, “A spatial hierarchical reasoning network for remote sensing visual question answering,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–15, 2023.
- [184] X. Dong, P. Zhang, Y. Zang, Y. Cao, B. Wang, L. Ouyang, X. Wei, S. Zhang, H. Duan, M. Cao, W. Zhang, Y. Li, H. Yan, Y. Gao, X. Zhang, W. Li, J. Li, K. Chen, C. He, X. Zhang, Y. Qiao, D. Lin, and J. Wang, “Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model,” *arXiv preprint arXiv:2401.16420*, 2024.
- [185] M. Zhang, F. Chen, and B. Li, “Multi-step question-driven visual question answering for remote sensing,” *IEEE Transactions on Geoscience and Remote Sensing*, 2023.
- [186] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, “Joint 2d-3d-semantic data for indoor scene understanding,” *arXiv preprint arXiv:1702.01105*, 2017.

- [187] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, “The replica dataset: A digital replica of indoor spaces,” *arXiv preprint arXiv:1906.05797*, 2019.
- [188] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- [189] Y. Wu, A. Kirillov, and F. Massa, “W.-y. lo i r. girshick, detectron2,” 2019.
- [190] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
- [191] OpenAI, :, A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, A. Madry, A. Baker-Whitcomb, A. Beutel, A. Borzunov, A. Carney, A. Chow, A. Kirillov, A. Nichol, A. Paino, A. Renzin, A. T. Passos, A. Kirillov, A. Christakis, A. Conneau, A. Kamali, A. Jabri, A. Moyer, A. Tam, A. Crookes, A. Tootoochian, A. Tootoonchian, A. Kumar, A. Val-lone, A. Karpathy, A. Braunstein, A. Cann, A. Codispoti, A. Galu, A. Kondrich, A. Tulloch, A. Mishchenko, A. Baek, A. Jiang, A. Pelisse, A. Woodford, A. Gosalia, A. Dhar, A. Pantuliano, A. Nayak, A. Oliver, B. Zoph, B. Ghorbani, B. Leimberger, B. Rossen, B. Sokolowsky, B. Wang, B. Zweig, B. Hoover, B. Samic, B. McGrew, B. Spero, B. Giertler, B. Cheng, B. Lightcap, B. Walkin, B. Quinn, B. Guarraci, B. Hsu, B. Kellogg, B. Eastman, C. Lugaresi, C. Wainwright, C. Bassin, C. Hudson, C. Chu, C. Nelson, C. Li, C. J. Shern, C. Conger, C. Barette, C. Voss, C. Ding, C. Lu, C. Zhang, C. Beaumont, C. Hallacy, C. Koch, C. Gibson, C. Kim, C. Choi, C. McLeavey, C. Hesse, C. Fischer, C. Winter, C. Czarnecki, C. Jarvis, C. Wei, C. Koumouzelis, D. Sherburn, D. Kappler, D. Levin, D. Levy, D. Carr, D. Farhi, D. Mely, D. Robinson, D. Sasaki, D. Jin, D. Valladares, D. Tsipras, D. Li, D. P. Nguyen, D. Findlay, E. Oiwoh, E. Wong, E. Asdar, E. Proehl, E. Yang, E. Antonow, E. Kramer, E. Peterson, E. Sigler, E. Wal-lace, E. Brevdo, E. Mays, F. Khorasani, F. P. Such, F. Raso, F. Zhang, F. von Lohmann, F. Sulit, G. Goh, G. Oden, G. Salmon, G. Starace, G. Brockman, H. Salman, H. Bao, H. Hu, H. Wong, H. Wang, H. Schmidt, H. Whitney, H. Jun, H. Kirchner, H. P. de Oliveira Pinto, H. Ren, H. Chang, H. W. Chung, I. Kivlichan, I. O’Connell, I. O’Connell, I. Osband, I. Silber, I. Sohl, I. Okuyucu, I. Lan, I. Kostrikov, I. Sutskever, I. Kanitscheider, I. Gulrajani, J. Coxon, J. Menick, J. Pachocki, J. Aung, J. Betker, J. Crooks, J. Lennon, J. Kiros, J. Leike, J. Park, J. Kwon, J. Phang, J. Teplitz, J. Wei, J. Wolfe, J. Chen, J. Harris, J. Varavva, J. G. Lee, J. Shieh, J. Lin, J. Yu, J. Weng, J. Tang, J. Yu, J. Jang, J. Q. Candela, J. Beutler, J. Landers, J. Parish, J. Heidecke, J. Schulman, J. Lachman, J. McKay, J. Uesato, J. Ward, J. W. Kim, J. Huizinga, J. Sitkin, J. Kraaijeveld, J. Gross, J. Kaplan, J. Snyder, J. Achiam, J. Jiao, J. Lee, J. Zhuang, J. Harriman, K. Fricke, K. Hayashi, K. Singhal, K. Shi,

- K. Karthik, K. Wood, K. Rimbach, K. Hsu, K. Nguyen, K. Gu-Lemberg, K. Button, K. Liu, K. Howe, K. Muthukumar, K. Luther, L. Ahmad, L. Kai, L. Itow, L. Workman, L. Pathak, L. Chen, L. Jing, L. Guy, L. Fedus, L. Zhou, L. Mamitsuka, L. Weng, L. McCallum, L. Held, L. Ouyang, L. Feuvrier, L. Zhang, L. Kondraciuk, L. Kaiser, L. Hewitt, L. Metz, L. Doshi, M. Aflak, M. Simens, M. Boyd, M. Thompson, M. Dukhan, M. Chen, M. Gray, M. Hudnall, M. Zhang, M. Aljube, M. Litwin, M. Zeng, M. Johnson, M. Shetty, M. Gupta, M. Shah, M. Yatbaz, M. J. Yang, M. Zhong, M. Glaese, M. Chen, M. Janner, M. Lampe, M. Petrov, M. Wu, M. Wang, M. Fradin, M. Pokrass, M. Castro, M. O. T. de Castro, M. Pavlov, M. Brundage, M. Wang, M. Khan, M. Murati, M. Bavarian, M. Lin, M. Yesildal, N. Soto, N. Gimelshein, N. Cone, N. Staudacher, N. Summers, N. LaFontaine, N. Chowdhury, N. Ryder, N. Stathas, N. Turley, N. Tezak, N. Felix, N. Kudige, N. Keskar, N. Deutsch, N. Bundick, N. Puckett, O. Nachum, O. Okelola, O. Boiko, O. Murk, O. Jaffe, O. Watkins, O. Godement, O. Campbell-Moore, P. Chao, P. McMillan, P. Belov, P. Su, P. Bak, P. Bakkum, P. Deng, P. Dolan, P. Hoeschele, P. Welinder, P. Tillet, P. Pronin, P. Tillet, P. Dhariwal, Q. Yuan, R. Dias, R. Lim, R. Arora, R. Troll, R. Lin, R. G. Lopes, R. Puri, R. Miyara, R. Leike, R. Gaubert, R. Zamani, R. Wang, R. Donnelly, R. Honsby, R. Smith, R. Sahai, R. Ramchandani, R. Huet, R. Carmichael, R. Zellers, R. Chen, R. Chen, R. Nigmatullin, R. Cheu, S. Jain, S. Altman, S. Schoenholz, S. Toizer, S. Miserendino, S. Agarwal, S. Culver, S. Ethersmith, S. Gray, S. Grove, S. Metzger, S. Hermani, S. Jain, S. Zhao, S. Wu, S. Jomoto, S. Wu, Shuaiqi, Xia, S. Phene, S. Papay, S. Narayanan, S. Coffey, S. Lee, S. Hall, S. Balaji, T. Broda, T. Stramer, T. Xu, T. Gogineni, T. Christianson, T. Sanders, T. Patwardhan, T. Cunningham, T. Degry, T. Dimson, T. Raoux, T. Shadwell, T. Zheng, T. Underwood, T. Markov, T. Sherbakov, T. Rubin, T. Stasi, T. Kaftan, T. Heywood, T. Peterson, T. Walters, T. Eloundou, V. Qi, V. Moeller, V. Monaco, V. Kuo, V. Fomenko, W. Chang, W. Zheng, W. Zhou, W. Manassra, W. Sheu, W. Zaremba, Y. Patil, Y. Qian, Y. Kim, Y. Cheng, Y. Zhang, Y. He, Y. Zhang, Y. Jin, Y. Dai, and Y. Malkov, “Gpt-4o system card,” 2024.
- [192] OpenAI, :, A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, A. Iftimie, A. Karpenko, A. T. Passos, A. Neitz, A. Prokofiev, A. Wei, A. Tam, A. Bennett, A. Kumar, A. Saraiva, A. Vallone, A. Duberstein, A. Kondrich, A. Mishchenko, A. Applebaum, A. Jiang, A. Nair, B. Zoph, B. Ghorbani, B. Rossen, B. Sokolowsky, B. Barak, B. McGrew, B. Minaiev, B. Hao, B. Baker, B. Houghton, B. McKinzie, B. Eastman, C. Lugaresi, C. Bassin, C. Hudson, C. M. Li, C. de Bourcy, C. Voss, C. Shen, C. Zhang, C. Koch, C. Orsinger, C. Hesse, C. Fischer, C. Chan, D. Roberts, D. Kappler, D. Levy, D. Selsam, D. Dohan, D. Farhi, D. Mely, D. Robinson, D. Tsipras, D. Li, D. Oprica, E. Freeman, E. Zhang, E. Wong, E. Proehl, E. Cheung, E. Mitchell, E. Wallace, E. Ritter, E. Mays, F. Wang, F. P. Such, F. Raso, F. Leoni, F. Tsimpouras, F. Song, F. von Lohmann, F. Sulit, G. Salmon, G. Parascandolo, G. Chabot, G. Zhao, G. Brockman,

- G. Leclerc, H. Salman, H. Bao, H. Sheng, H. Andrin, H. Bagherinezhad, H. Ren, H. Lightman, H. W. Chung, I. Kivlichan, I. O’Connell, I. Osband, I. C. Gilaberte, I. Akkaya, I. Kostrikov, I. Sutskever, I. Kofman, J. Pachocki, J. Lennon, J. Wei, J. Harb, J. Twore, J. Feng, J. Yu, J. Weng, J. Tang, J. Yu, J. Q. Candela, J. Palermo, J. Parish, J. Heidecke, J. Hallman, J. Rizzo, J. Gordon, J. Uesato, J. Ward, J. Huizinga, J. Wang, K. Chen, K. Xiao, K. Singhal, K. Nguyen, K. Cobbe, K. Shi, K. Wood, K. Rimbach, K. Gu-Lemberg, K. Liu, K. Lu, K. Stone, K. Yu, L. Ahmad, L. Yang, L. Liu, L. Maksin, L. Ho, L. Fedus, L. Weng, L. Li, L. McCallum, L. Held, L. Kuhn, L. Kondraciuk, L. Kaiser, L. Metz, M. Boyd, M. Trebacz, M. Joglekar, M. Chen, M. Tintor, M. Meyer, M. Jones, M. Kaufer, M. Schwarzer, M. Shah, M. Yatbaz, M. Y. Guan, M. Xu, M. Yan, M. Glaese, M. Chen, M. Lampe, M. Malek, M. Wang, M. Fradin, M. McClay, M. Pavlov, M. Wang, M. Wang, M. Murati, M. Bavarian, M. Rohaninejad, N. McAleese, N. Chowdhury, N. Chowdhury, N. Ryder, N. Tezak, N. Brown, O. Nachum, O. Boiko, O. Murk, O. Watkins, P. Chao, P. Ashbourne, P. Izmailov, P. Zhokhov, R. Dias, R. Arora, R. Lin, R. G. Lopes, R. Gaon, R. Miyara, R. Leike, R. Hwang, R. Garg, R. Brown, R. James, R. Shu, R. Cheu, R. Greene, S. Jain, S. Altman, S. Toizer, S. Toyer, S. Miserendino, S. Agarwal, S. Hernandez, S. Baker, S. McKinney, S. Yan, S. Zhao, S. Hu, S. Santurkar, S. R. Chaudhuri, S. Zhang, S. Fu, S. Papay, S. Lin, S. Balaji, S. Sanjeev, S. Sidor, T. Broda, A. Clark, T. Wang, T. Gordon, T. Sanders, T. Patwardhan, T. Sottiaux, T. Degry, T. Dimson, T. Zheng, T. Garipov, T. Stasi, T. Bansal, T. Creech, T. Peterson, T. Eloundou, V. Qi, V. Kosaraju, V. Monaco, V. Pong, V. Fomenko, W. Zheng, W. Zhou, W. McCabe, W. Zaremba, Y. Dubois, Y. Lu, Y. Chen, Y. Cha, Y. Bai, Y. He, Y. Zhang, Y. Wang, Z. Shao, and Z. Li, “Openai o1 system card,” 2024.
- [193] G. Bertasius, H. Wang, and L. Torresani, “Is space-time attention all you need for video understanding?” 2021.
- [194] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” 2021.
- [195] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer, “Videoclip: Contrastive pre-training for zero-shot video-text understanding,” 2021.
- [196] A. V. Etten, D. Lindenbaum, and T. M. Bacastow, “Spacenet: A remote sensing dataset and challenge series,” 2019.
- [197] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, “A large contextual dataset for classification, detection and counting of cars with deep learning,” 2016.
- [198] X. Li, A. Yuan, and X. Lu, “Multi-modal gated recurrent units for image description,” *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29847–29869, 2018.



- [199] K. Xu, “Show, attend and tell: Neural image caption generation with visual attention,” *arXiv preprint arXiv:1502.03044*, 2015.
- [200] C. Liu, R. Zhao, and Z. Shi, “Remote-sensing image captioning based on multilayer aggregated transformer,” *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

# Appendix A

## Extension of Battery Operation

Lithium metal batteries (LMBs) have the potential to offer higher energy density and improved efficiency compared to traditional lithium-ion batteries (LIBs). This appendix outlines the basic working principles of lithium batteries in general, including the movement of lithium ions and electrons during charge and discharge, the role of battery components, the formation of the solid electrolyte interface (SEI), and some challenges, such as dendrite formation.

- Appendix A.1 provides an overview of the components of lithium batteries.
- Appendix A.2 explains how lithium-ion batteries operate.
- Appendix A.3 explains how lithium metal batteries operate.
- Appendix A.4 discusses the motivation for moving from lithium-ion batteries to lithium metal batteries.
- Appendix A.5 explains dendrite formation and its risks.

### A.1 Overview of Lithium-Ion and Lithium-Metal Batteries

Lithium-Ion and Lithium-Metal typically consist of the following components:

- **Current Collectors:** For both LIBs and LMBs, current collectors facilitate electron collection and distribution at the electrodes. Copper is commonly used on the anode side, while aluminum is used on the cathode side.
- **Anode:** For LMBs, the anode is made of lithium metal, which has a high theoretical capacity of 3,860 mAh/g. This is a key difference from LIBs, where the anode is usually made of graphite, which has a theoretical capacity of 372 mAh/g.

- **Cathode:** LIBs typically use lithium metal oxide cathodes, such as Lithium Nickel Manganese Cobalt Oxide (NMC), Lithium Iron Phosphate (LFP), Lithium Manganese Oxide (LMO), Lithium Nickel Cobalt Aluminum Oxide (NCA), or Lithium Cobalt Oxide ( $\text{LiCoO}_2$ ), paired with a graphite-based anode. In contrast, LMBs differ primarily by using a pure lithium metal anode, and can employ similar lithium transition metal oxide cathodes, including NMC, NCA, and  $\text{LiCoO}_2$ . Additionally, some LMB designs incorporate alternative cathode materials, such as sulfur (used in lithium-sulfur batteries) and conversion-type compounds like iron fluoride ( $\text{FeF}_3$ ) and copper fluoride ( $\text{CuF}_2$ ), which offer the potential for higher energy densities.
- **Electrolyte:** The electrolyte, which can be in liquid, gel, or solid form, allows lithium ions to move between the anode and cathode. In liquid electrolytes, this typically consists of lithium salts dissolved in organic solvents, while gel and solid electrolytes use different materials, such as polymers or ceramics, to conduct lithium ions.
- **Separator:** The separator is a porous material placed between the anode and cathode to allow the passage of lithium ions but prevent electrical contact between the electrodes, which would lead to short circuits.

## A.2 How Lithium-Ion Batteries Work

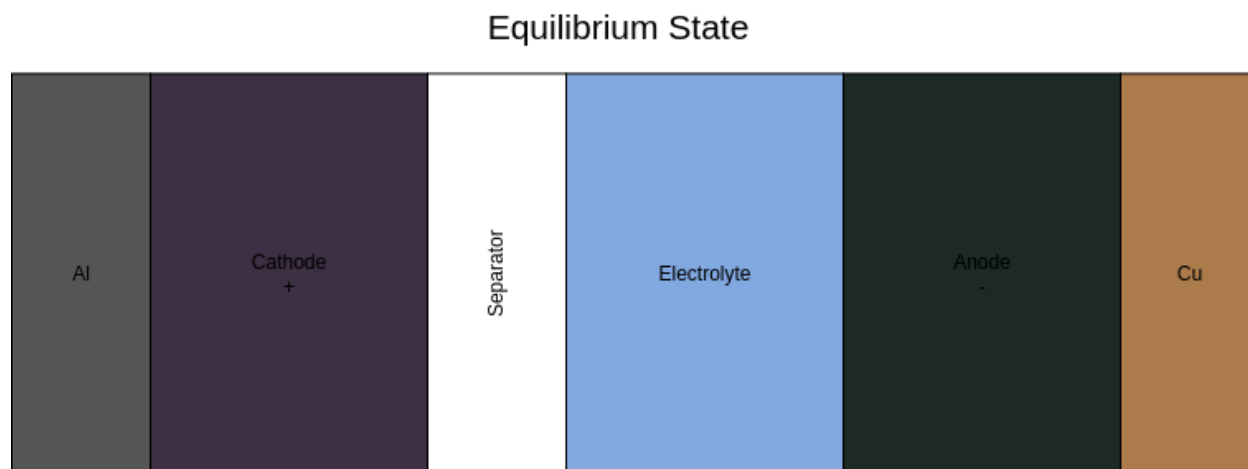


Figure A.1: Diagram of a battery at equilibrium.

In LIBs, lithium is stored at the anode as neutral lithium atoms intercalated between the layers of carbon atoms in graphite at equilibrium as shown in Figure A.1. When the battery is discharging and supplying power to a device as depicted on Figure A.2, these lithium atoms are oxidized at the anode. This means each lithium atom loses an electron, becoming

a lithium ion ( $\text{Li}^+$ ). The oxidizing agent in this case is the external circuit, which provides a low-resistance path for the electrons to flow toward the cathode, driven by the electrical potential difference between the electrodes.

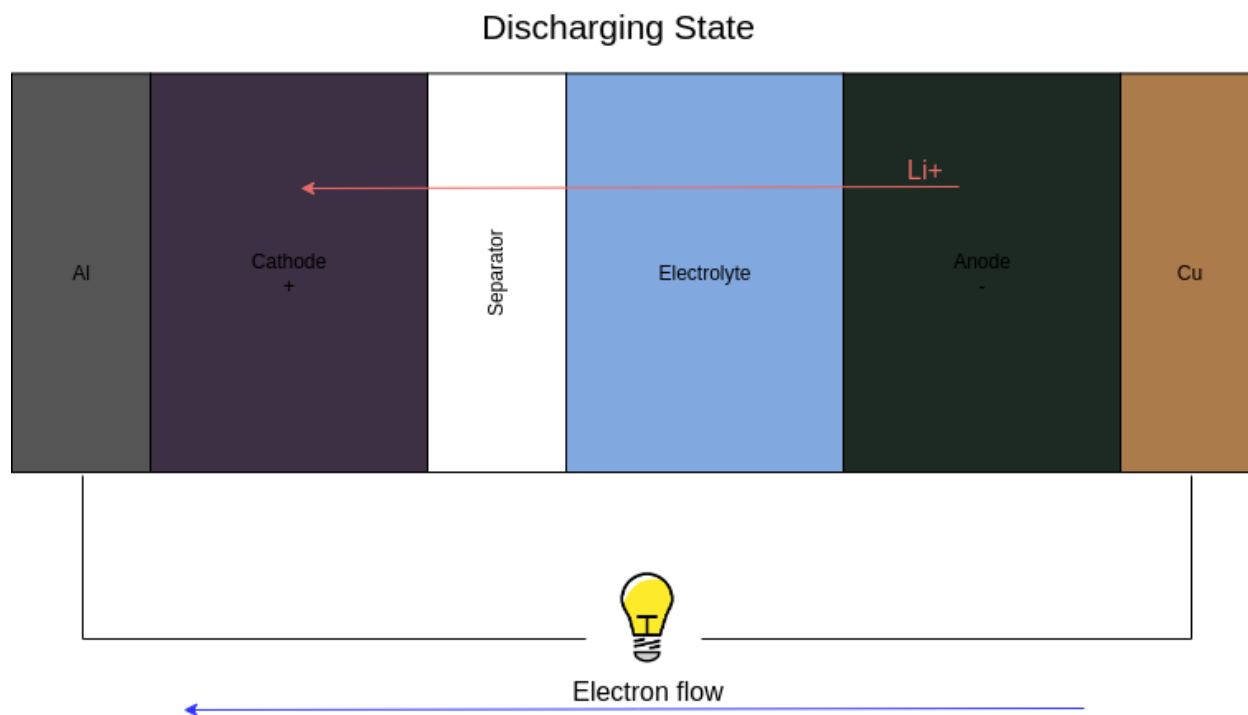


Figure A.2: Diagram of a battery during its discharging state.

As electrons move through the external load, the lithium ions travel through the electrolyte and the porous polymer separator toward the cathode to balance the negative charge buildup at the cathode. As mentioned above, the cathode typically consists of a lithium metal oxide, such as lithium cobalt oxide ( $\text{LiCoO}_2$ ), where the transition metal ions (like cobalt) are reduced by the incoming electrons. Lithium ions are simultaneously intercalated into the cathode structure to maintain charge neutrality.

This process continues until most of the usable lithium has moved from the anode to the cathode, at which point the battery is considered discharged. When charging as shown in Figure A.3, an external power source applies a voltage that forces electrons to move from the cathode back to the anode. Lithium ions also migrate back across the electrolyte and separator to the anode, where they recombine with the returning electrons and are once again intercalated between the graphite layers, restoring the battery to a charged state.

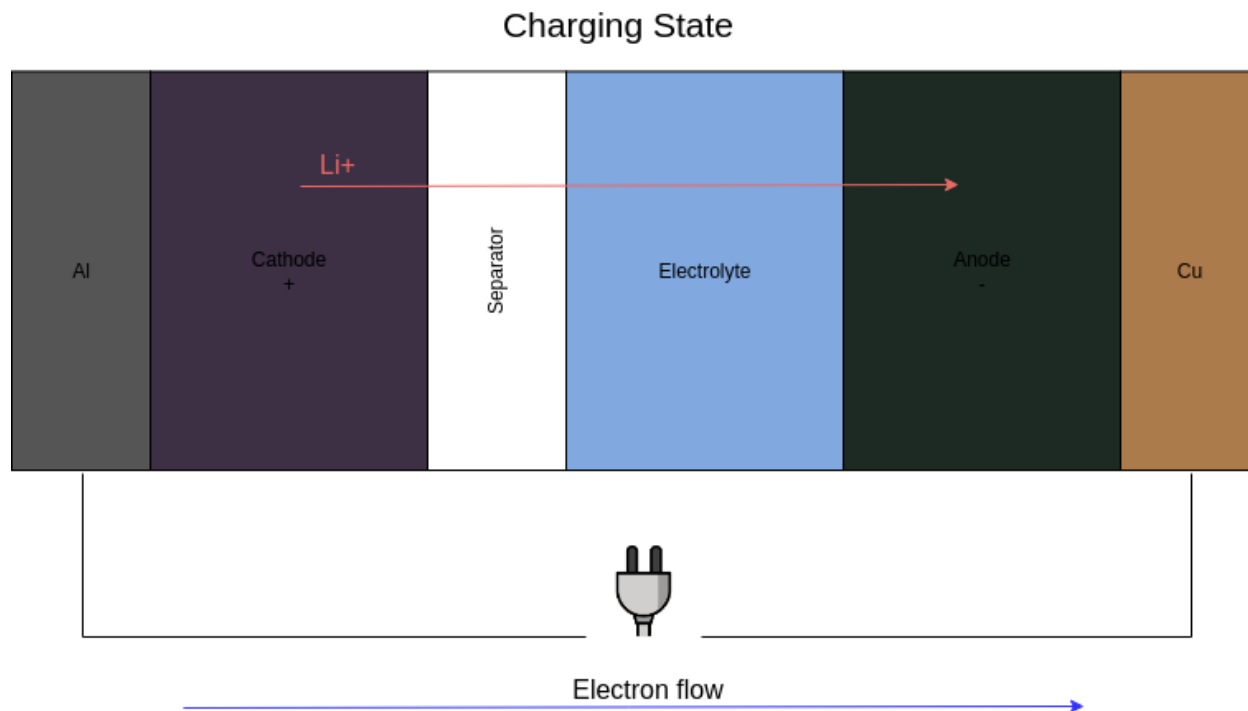


Figure A.3: Diagram of a battery during its charging state.

### A.3 How Lithium Metal Batteries Work

In LMBs, the anode is not graphite but pure lithium metal. During discharge, lithium metal at the anode is oxidized, releasing lithium ions into the electrolyte and electrons into the external circuit. The cathode may be a lithium metal oxide, sulfur, or a conversion-type material such as iron fluoride. At the cathode, the lithium ions are reduced and either intercalated, alloyed, or react to form new compounds, depending on the cathode chemistry. During charging, an external power source drives lithium ions back toward the anode, where they are reduced and deposited as metallic lithium. This direct lithium plating enables significantly higher theoretical energy density compared to LIBs, but it also introduces challenges such as dendrite formation, which can cause internal short circuits and pose safety risks.

## A.4 Why Move from Lithium-Ion Batteries to Lithium Metal Batteries

In LIBs, the electrolyte is crucial for enabling lithium ions to safely move back and forth between the anode and cathode during charging and discharging. The anode, typically made of graphite as mentioned above, is thermodynamically unstable when in contact with the electrolyte. This instability leads to the formation of a Solid-Electrolyte Interphase (SEI) layer on the surface of the anode during the initial cycles of the battery. The SEI acts as a protective barrier, preventing direct contact between the electrolyte and the intercalated lithium in the graphite, which helps suppress further side reactions that could degrade the battery’s performance.

When the battery reaches elevated temperatures, typically around 70 degrees Celsius, and the heat is not dissipated, the SEI layer can begin to decompose. This decomposition allows more extensive reactions between the anode and the electrolyte, generating additional heat. If the temperature continues to rise, the polymer separator can begin to degrade or melt, potentially allowing direct contact between the anode and cathode. This internal short circuit accelerates exothermic reactions, causing the temperature to rise even further. Eventually, the cathode material, such as lithium cobalt oxide, may decompose and release oxygen, which can react with flammable electrolyte components and lead to fire or explosion. This self-reinforcing sequence of events is known as *thermal runaway* and is one of the primary safety risks associated with LIBs.

In addition to addressing safety concerns, the motivations for the development of LMBs include the potential for significantly higher energy density and the goal of enabling faster charging, as illustrated in Table 2.2. Replacing the graphite anode with lithium metal increases the theoretical storage capacity by both weight and volume, which can extend the range and improve performance in high-energy-demand applications. Although fast charging is a key research objective for LMBs, it remains challenging due to issues such as lithium dendrite formation, which can compromise both safety and cycle life. These challenges motivate our investigation of dendrite morphology in chapter 2, with the aim of informing the design of safer and more efficient lithium metal batteries.

## A.5 Dendrite Formation and its Risks

A key challenge in lithium metal batteries is the formation of dendrites. When the battery is charged, lithium ions deposit onto the anode. If the deposition is uneven, dendrites can form needle-like structures of lithium metal as shown in Figure A.4 from (5). These dendrites can grow through the electrolyte and cause short circuits, potentially leading to battery failure or even thermal runaway. Efforts are underway to develop methods that control lithium deposition, ensuring even plating and reducing the risk of dendrite growth. Hence, our proposed method to understand dendrites’ morphology contributes to building safer and more powerful batteries.

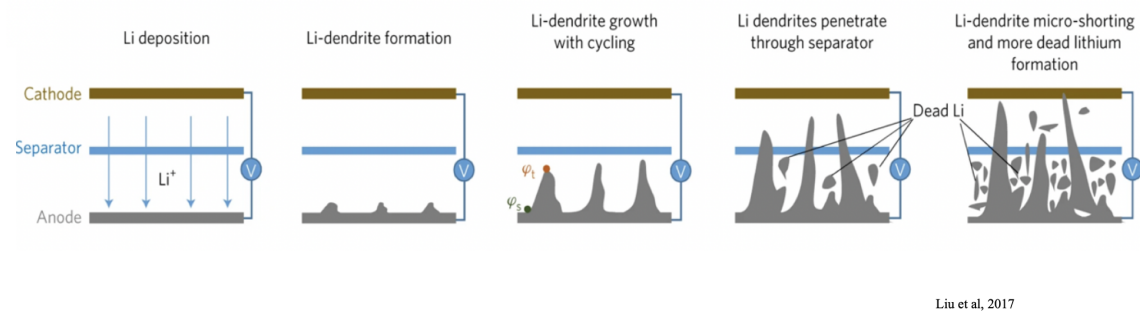


Figure A.4: Dendrite formation in lithium metal batteries as shown in (5).

# Appendix B

## Extension of Y-Net

This appendix highlights the robustness, generalization, and versatility of the Y-Net architecture through two key demonstrations. Y-Net’s ability to produce accurate segmentation, even under incomplete supervision, shows the effectiveness of its dual-branch design and residual feature refinement strategy. Additionally, we qualitatively demonstrate that Y-Net extends beyond segmentation to perform accurate depth and uncertainty estimation, showcasing its potential as a flexible, general-purpose vision backbone. Together, these examples illustrate the strength of Y-Net in addressing challenging visual tasks across diverse scenarios.

- Appendix B.1 shows Y-Net correctly predicting segmentation masks even in areas with missing ground truth annotations.
- Appendix B.2 depicts Y-Net’s successful adaptation to monocular depth and uncertainty estimation in equirectangular imagery, further demonstrating its architectural flexibility.

### B.1 Y-Net Robustness and Generalization

Y-Net exhibits remarkable robustness and generalization in segmentation tasks, particularly in scenarios where ground truth labels are incomplete or missing, as shown in Figure B.1. This capability stems from its dual-branch architecture, which fuses local and global feature extraction through the Regular Convolutional Module and the right-branch combination of Dilated Convolution and Pyramid Pooling Modules. By capturing both fine-grained textures and long-range contextual information, Y-Net can infer semantic structure even in ambiguous regions. Residual connections between branches allow the network to self-correct and reinforce weak signals, resulting in accurate predictions that often extend beyond the annotated ground truth. Empirical observations such as Figure B.1 confirm that Y-Net can identify valid regions omitted in GT labels, suggesting its learned representations are resilient to annotation noise and capable of meaningful generalization.



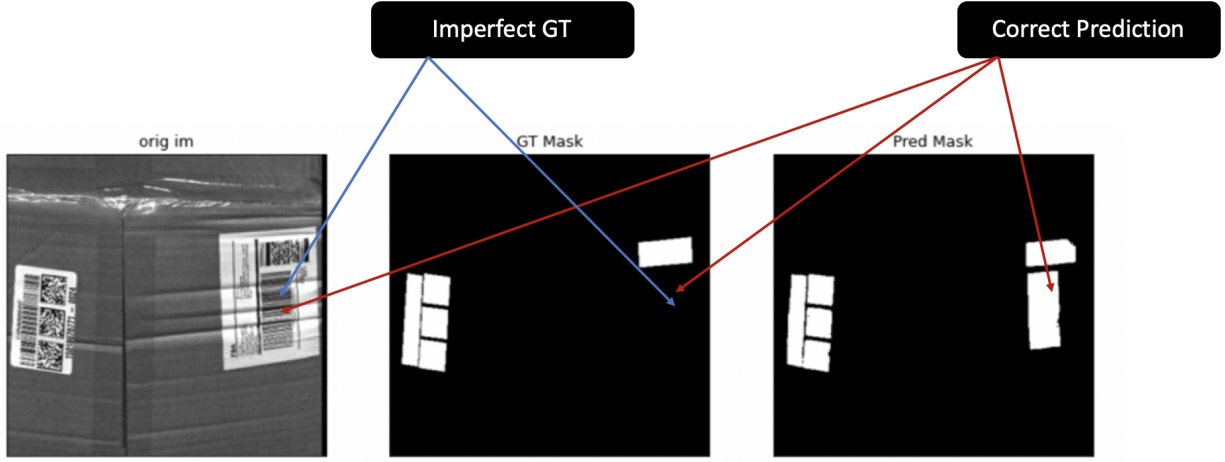


Figure B.1: Y-Net segmentation predictions extending beyond incomplete ground truth annotations.

## B.2 Y-Net for Depth and Uncertainty Estimation

In this section, we present the adaptation of the Y-Net architecture to two complementary computer vision tasks: monocular depth estimation and aleatoric uncertainty estimation. Originally designed for barcode segmentation, Y-Net was modified to include dual-task output heads. We found that its encoder-decoder architecture is inherently well-suited for simultaneous regression and dense prediction. By leveraging this flexibility, we extended the model to predict both the depth map of a scene and the associated aleatoric uncertainty within a unified framework.

Accurate depth estimation from single RGB images is crucial for a range of applications, including autonomous navigation, 3D reconstruction, and robotic manipulation. One particularly compelling application is drone-based navigation, where lightweight, cost-effective sensors are critical, and monocular cameras are often the only available modality due to payload constraints. In such scenarios, equirectangular RGB images captured using wide field-of-view or panoramic cameras offer a holistic, 360-degree scene representation, making them well-suited for obstacle avoidance and situational awareness in complex environments.

However, predicting depth from these monocular images is fundamentally ill-posed and subject to ambiguity, especially in textureless regions or areas with occlusions. To address this, our model also estimates aleatoric uncertainty, which captures data-dependent noise and reflects the inherent uncertainty in visual input. This uncertainty estimation is particularly helpful in drone navigation, as it enables the system to reason probabilistically about prediction confidence and take precautionary actions in uncertain regions of the scene to avoid collision.

For the dataset, we use the Stanford 2D-3D-S dataset (186), which provides 1,413

equirectangular RGB images along with their corresponding depth maps. In addition, we generate an extended dataset comprising 98,000 equirectangular RGB-depth pairs using the Replica dataset (187). The Replica dataset comes with 18 highly photorealistic 3D indoor scene reconstructions at both room and building scales, spanning a variety of environments such as six different configurations of the Facebook Reality Labs (FRL) apartment, five office rooms, a two-floor house, two multi-room apartment spaces, a hotel room, and three individual apartment rooms. To generate our dataset, we develop a script that randomly places a virtual 360° camera at valid positions within randomly selected locations and captures both the rendered equirectangular RGB image and its corresponding depth map.

We retain the dual-head configuration of Y-Net mentioned above, modifying the output heads to support the following:

- Depth Prediction Decoder: Outputs a dense, per-pixel depth map.
- Uncertainty Estimation Decoder: Outputs a dense map of the per-pixel log-variance ( $\log \sigma^2$ ), following the modeling approach introduced by (188).

The model is trained using the heteroscedastic uncertainty loss function Equation B.1 proposed by (188), which jointly optimizes for depth accuracy and uncertainty regularization:

$$\mathcal{L}_N(\theta) = \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|^2 + \frac{1}{2} \log \sigma(x_i)^2 \right) \quad (\text{B.1})$$

Where:

- $y_i$  is the ground-truth depth at pixel  $i$ ,
- $f(x_i)$  is the predicted depth from input  $x_i$ ,
- $\sigma(x_i)^2$  is the predicted variance at pixel  $i$ ,
- $N$  is the number of valid pixels,
- $\theta$  are the model parameters.

Training is conducted using the Adam optimizer with a constant learning rate of 0.004 and weight decay of  $10^{-4}$  with early stopping based on validation loss. We also perform data augmentation with standard techniques such as horizontal flipping and color jittering.

Qualitative results for (186) are shown in Figure B.2 and those for (187) are shown in Figure B.3. As can be seen, the adapted Y-Net achieves competitive performance in depth estimation while also producing meaningful uncertainty maps.

This analysis reveals that high uncertainty values correlate with object boundaries and primarily with geometric features such as locations with strong curvature in the equirectangular images, reflective surfaces, and occluded regions, which is consistent with areas of ambiguous visual information.

This demonstrates that the Y-Net architecture can be effectively repurposed for depth and aleatoric uncertainty estimation with minimal architectural changes. The resulting model produces interpretable and accurate outputs, making it well-suited for deployment in real-world perception systems where understanding prediction confidence is as important as the prediction itself.

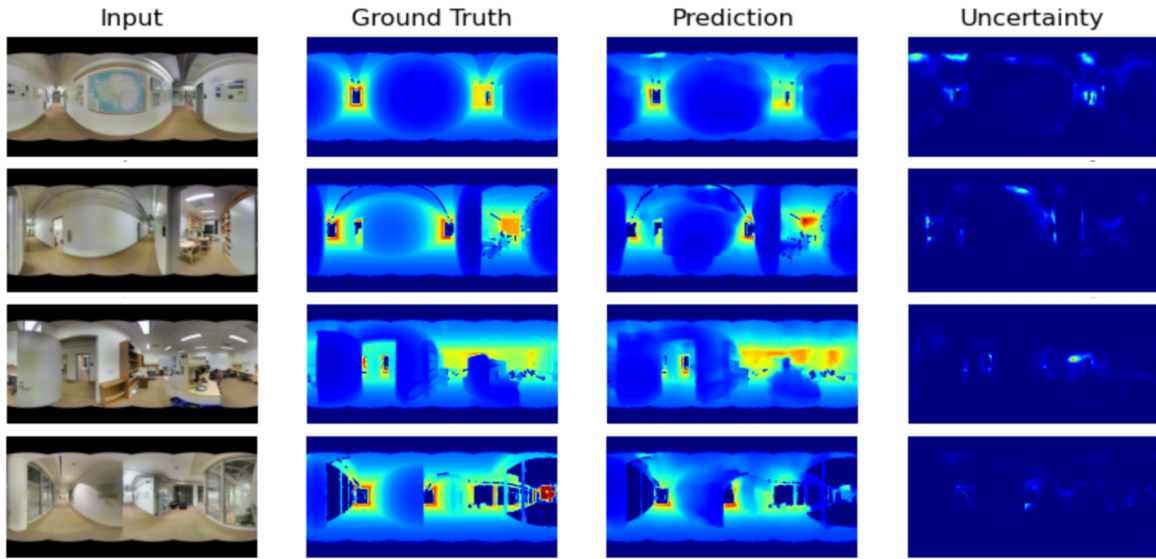


Figure B.2: Qualitative results of Y-Net on depth and uncertainty estimation for the Stanford 2D-3D-S dataset.

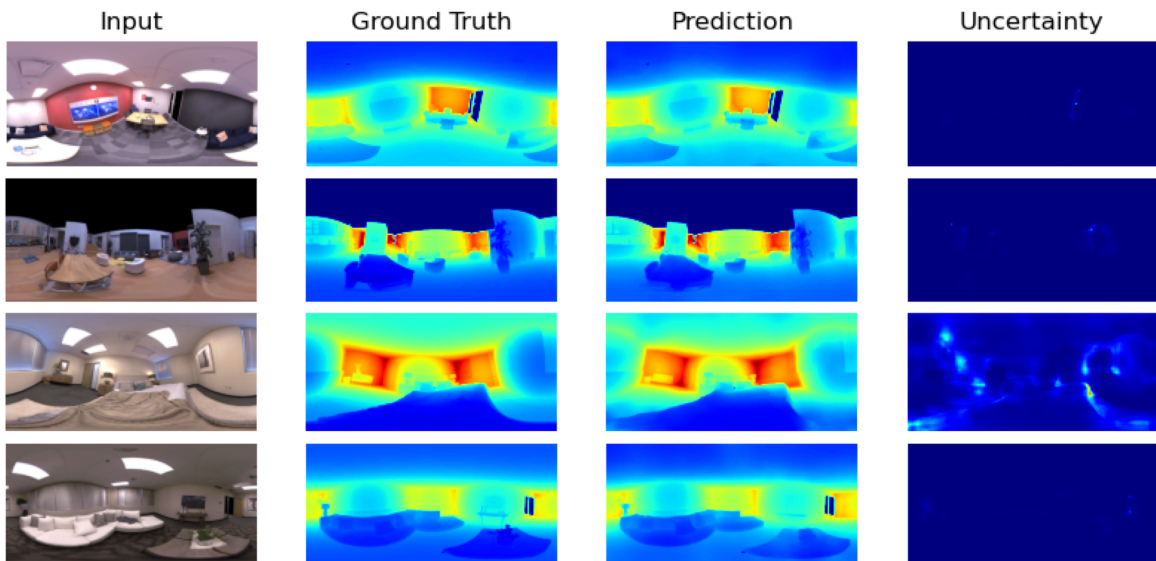


Figure B.3: Qualitative results of Y-Net on depth and uncertainty estimation for the Replica dataset.

# Appendix C

## Training Multimodal Models on High Performance Computers (HPC)

This appendix provides sample scripts used for training our models on large datasets across multiple nodes of a high-performance computing (HPC) cluster using SLURM (Simple Linux Utility for Resource Management). The system consists of multiple interconnected computing nodes, each equipped with its own CPU, memory, and sometimes GPU resources. These nodes are linked via a high-speed network, enabling the distribution of computational tasks across multiple machines.

- Appendix C.1 details the code release, including links to the codebases and datasets used in this project.
- Appendix C.2 shows the script used while launching the training manually with `Pytorch DDP` or `FSDP`.
- Appendix C.3 allows automatic relaunch of training with `Pytorch DDP` or `FSDP` when it crashes unexpectedly due to system issues.
- Appendix C.4 shows of part of the script that needs to be updated for `DeepSeep`.
- Appendix C.5 presents more details on the OXE and LLARVA dataset.

### C.1 Code Release

Details on licensing, user consent, and personally identifiable information (PII) for each dataset are available in their corresponding original publications. We note that the datasets used in this work are commonly adopted in related research and do not include any offensive or harmful material. As with any machine learning system, responsible use is advised. More information about our project can be found [here](#).

## C.2 Manual Launch Script

```
#!/bin/bash

# Parameters
#SBATCH --output=path/to/slurm_dir/train_${SLURM_JOB_NUM_NODES}
#          _nodes_rpt_%N/%j/%j_0_log.out
#SBATCH --error=path/to/slurm_dir/train_${SLURM_JOB_NUM_NODES}
#          _nodes_rpt_%N/%j/%j_0_log.err
#SBATCH --account=$ACCOUNT_NAME
#SBATCH --qos=$QOS_NAME
#SBATCH --job-name=$MODEL_NAME
#SBATCH --open-mode=append
#SBATCH --signal=$SIGNAL_NAME
#SBATCH --time=168:00:00
#SBATCH --constraint=$CONSTRAINT_NAME
#SBATCH --nodes=$NUM_NODES
#SBATCH --ntasks=$NUM_TASKS
#SBATCH --gres=gpu:NUM_GPU_NAME:$NUM_GPU_PER_NODE
#SBATCH --mail-type=ALL
#SBATCH --mail-user=$USER_EMAIL
#SBATCH --export=ALL
#SBATCH --propagate=STACK

ulimit -s unlimited

create_folder_if_not_exists() {
    local folder_name="$1"
    if [ ! -d "$folder_name" ]; then
        mkdir "$folder_name"
        echo "Folder '$folder_name' created."
    else
        echo "Folder '$folder_name' already exists."
    fi
}

# setup
export EXP_NAME="rpt"
PROJ_ROOT="${HOME}/Projects/${EXP_NAME}"
ENV_NAME=${EXP_NAME}

OUTPUT_DIR="${WORKDIR}/outputs/$MODEL_NAME/training_dir_${SLURM_JOB_NUM_NODES}_nodes_${BC_HOST}/${SLURM_JOB_ID}"
LOG_DIR="${OUTPUT_DIR}/logs"

echo "PROJ_ROOT ${PROJ_ROOT}"
echo "OUTPUT_DIR ${OUTPUT_DIR}"
echo "LOG_DIR ${LOG_DIR}"

create_folder_if_not_exists "$OUTPUT_DIR"
create_folder_if_not_exists "$LOG_DIR"

pushd "${PROJ_ROOT}"
cp "${PROJ_ROOT}/example_launch_scripts/manual.slurm" "${LOG_DIR}/"

# Source conda env
if [[ -d "${HOME}/mambaforge" ]]; then
    CONDA_FN="mamba"
    CONDA_DIR="${HOME}/mambaforge"
```

```

elif [[ -d "${HOME}/anaconda3" ]]; then
    CONDA_FN="conda"
    CONDA_DIR="${HOME}/anaconda3"
elif [[ -d "${HOME}/miniconda3" ]]; then
    CONDA_FN="conda"
    CONDA_DIR="${HOME}/miniconda3"
fi

echo "CONDA_FN: $CONDA_FN"
echo "CONDA_DIR: $CONDA_DIR"

if [ -d "${CONDA_DIR}/etc/profile.d" ]; then
    source "${CONDA_DIR}/etc/profile.d/conda.sh"
fi
if [ -f "${CONDA_DIR}/etc/profile.d/mamba.sh" ]; then
    source "${CONDA_DIR}/etc/profile.d/mamba.sh"
fi

$CONDA_FN activate "${ENV_NAME}"
$CONDA_FN info --envs

export NGPUS_TOTAL=${SLURM_NTASKS}
export MASTER_PORT=29500
export WORLD_SIZE="${NGPUS_TOTAL}"

echo "NODELIST=${SLURM_NODELIST}"
master_addr=$(scontrol show hostnames "$SLURM_JOB_NODELIST" | head -n 1)
export MASTER_ADDR=$master_addr
echo "MASTER_ADDR=$MASTER_ADDR"

export HYDRA_FULL_ERROR=1
export OC_CAUSE=1

srun \
    --unbuffered \
    --output ${OUTPUT_DIR}/${j}_${t}_log.out \
    --error ${OUTPUT_DIR}/${j}_${t}_log.err \
    python -u \
        ${PROJ_ROOT}/tools/train_mma.py \
        num_gpus="${NGPUS_TOTAL}" \
        --config-name "mma/rptx/config_${MODEL_NAME}.yaml"

```

### C.3 Automatic Launch Script

```

import argparse
import os
import uuid
import submitit

from pathlib import Path
from omegaconf import OmegaConf

def parse_args():
    parser = argparse.ArgumentParser("Submitit for Segmentor Train+Val")
    parser.add_argument("--job_dir", default="", type=str, help="Job dir
        . Leave empty for automatic.")

```

```

parser.add_argument("--qos", default="", type=str, choices=(
    "frontier", "standard"), help="Queue to use")
parser.add_argument("--account", default="", type=str, help="The
    Account string to use")
parser.add_argument("--constraint", default="viz", type=str, help="
    Which Nautilus or Raider constraint to use")
parser.add_argument("--nodes", default=2, type=int, help="Number of
    nodes to request")
parser.add_argument("--config_path", default="/PATH/TO/MODEL_DIR/
    configs/config_${MODEL_NAME}.yaml", type=str, help="Path to
    config file")
parser.add_argument("--exclude", default="", type=str, help="Comma
    separated list of nodes to exclude")
parser.add_argument("--timeout", default=7*24*60, type=int, help="
    Duration of the job (min)")
parser.add_argument("--comment", default="", type=str, help="Comment
    to pass to scheduler")
parser.add_argument("--name", default="rpt", type=str, help="
    experiment name")
return parser.parse_args()

def get_shared_folder() -> Path:
    # user = os.getenv("USER")
    job_dir_path = Path(f'{os.environ["WORKDIR"]}/outputs/${MODEL_NAME}/
        training_dir_{os.environ["TOTAL_NUM_OF_NODES"]}nodes_{os.
        environ["BC_HOST"]}') # _{os.environ["SLURM_JOB_ID"]} {os.
        environ["SLURM_JOB_ID"]} # BC_NODE_ALLOC
    if not os.path.exists(job_dir_path):
        os.makedirs(job_dir_path)
    if Path(job_dir_path).is_dir():
        return job_dir_path
    raise RuntimeError("No shared folder available")

def get_init_file():
    # Init file must not exist, but it's parent dir must exist.
    os.makedirs(str(get_shared_folder()), exist_ok=True)
    init_file = get_shared_folder() / f"{uuid.uuid4().hex}_init"
    if init_file.exists():
        os.remove(str(init_file))
    return init_file

class Trainer(object):
    def __init__(self, args):
        self.args = args

    def __call__(self):
        self._setup_gpu_args()
        import TRAINER_FILE as trainer

        cfg = OmegaConf.load(self.args.config_path)
        cfg.num_gpus = int(cfg.num_nodes) * int(cfg.num_gpus_per_nodes)

        self.args.config = cfg
        trainer.train(cfg)

    def checkpoint(self):
        import os
        import submitit

```



```

# pass
self.args.dist_url = get_init_file().as_uri()
checkpoint_file = os.path.join(self.args.output_dir, "checkpoint
.pth")
if os.path.exists(checkpoint_file):
    self.args.resume = checkpoint_file
print("Requeueing ", self.args)
empty_trainer = type(self)(self.args)
return submitit.helpers.DelayedSubmission(empty_trainer)

def _setup_gpu_args(self):
    import os
    from pathlib import Path
    import submitit

    job_env = submitit.JobEnvironment()
    dist_env = submitit.helpers.TorchDistributedEnvironment().export
    (
        set_cuda_visible_devices=False
    )
    self.args.output_dir = Path(str(self.args.output_dir).replace("%
j", str(job_env.job_id)))
    self.args.log_dir = self.args.output_dir
    self.args.gpu = job_env.local_rank
    self.args.rank = job_env.global_rank
    self.args.world_size = job_env.num_tasks

    print(f"master: {dist_env.master_addr}:{dist_env.master_port}")
    print(
        f"World size: {dist_env.world_size}, Process group: {job_env
.num_tasks} tasks, rank: {job_env.global_rank}"
    )

def main():
    slurm_args = parse_args()

    if slurm_args.job_dir == "":
        slurm_args.job_dir = get_shared_folder() / "%j"

    # Note that the folder will depend on the job_id, to easily track
    experiments
    executor = submitit.AutoExecutor(folder=slurm_args.job_dir,
        slurm_max_num_timeout=30)

    if slurm_args.constraint == "$CONSTRAINT_NAME_1":
        num_gpus_per_node = 4
    elif slurm_args.constraint == "CONSTRAINT_NAME_2":
        num_gpus_per_node = 1
    else:
        num_gpus_per_node = 0

    kwargs = {}
    if slurm_args.comment:
        kwargs['slurm_comment'] = slurm_args.comment

    executor.update_parameters(
        gpus_per_node=num_gpus_per_node,
        tasks_per_node=num_gpus_per_node, # one task per GPU
        nodes=slurm_args.nodes,

```

```

        timeout_min=slurm_args.timeout,
        slurm_qos=slurm_args.qos,
        slurm_signal_delay_s=120,
        slurm_account=slurm_args.account,
        slurm_constraint=slurm_args.constraint,
        slurm_exclude=slurm_args.exclude,
        name=slurm_args.name,
        **kwargs
    )

    slurm_args.dist_url = get_init_file().as_uri()
    slurm_args.output_dir = slurm_args.job_dir

    trainer = Trainer(slurm_args)
    job = executor.submit(trainer)

    print("Submitted job_id:", job.job_id)

if __name__ == "__main__":
    main()

```

## C.4 DeepSpeed Launch Script

```

#!/bin/sh

...

srun \\\
--unbuffered \\\
--output "${LOG_DIR}/%j/node%n-task%t-srun.out.log" \\\
--error "${LOG_DIR}/%j/node%n-task%t-srun.err.log" \\\
--jobid "${SLURM_JOBID}" \\\
python "${PROJ_ROOT}/path/to/train/train_mem.py" \\\
--deepspeed "${PROJ_ROOT}/scripts/zero3.json" \\\
--lora_enable "${LORA_ENABLE}" \\\
--lora_r "${LORA_R}" \\\
--lora_alpha "${LORA_ALPHA}" \\\
...

```

or

```

#!/bin/sh

...

srun \\\
--unbuffered \\\
--output "${LOG_DIR}/%j/node%n-task%t-srun.out.log" \\\
--error "${LOG_DIR}/%j/node%n-task%t-srun.err.log" \\\
--jobid ${SLURM_JOBID} \\\

```

```

deepspeed \
  --include "\${GPU_SETTINGS}" \
  --master_port="\${MASTER_PORT}" \
  "\${PROJ_ROOT}/training_file.py" \
  ...

```

## C.5 More on OXE and LLARVA Datasets

As depicted in (6), the Open-X-Embodiment distribution is shown in Figure C.1.

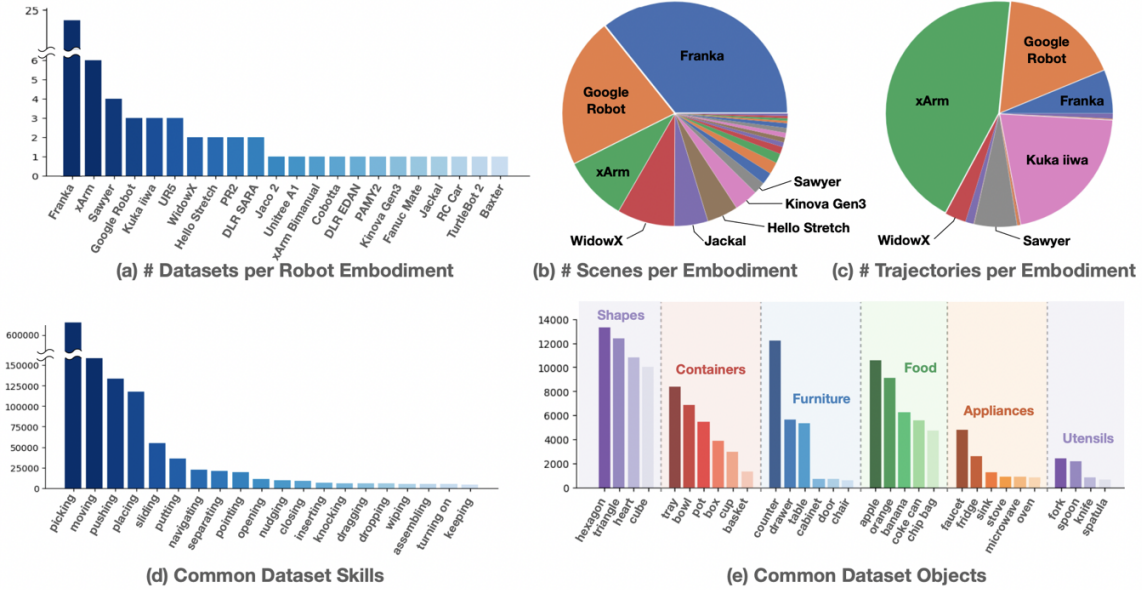


Figure C.1: Diagram of OXE dataset distribution as shown in (6); (a) shows the dataset distribution per robot; (b) shows that the Franka robot has the most diversity in distinct scenes; (c) shows that xArm and Google Robot have the most trajectories; (d, e) show the data distribution by skills and common objects.

The OXE (6) was used to generate 8.5 million image–visual trace pairs, as shown in Figure C.2. As referenced in (4), we used 13 different robots (shown in Table C.1), covering a variety of cameras, tasks, environments, and end-effectors. We obtained the 2-D visual traces for each image using the Detectron2 (189) implementation of Faster R-CNN (190), trained to detect all types of end-effectors present in (6). The center of the detected bounding box in each frame of a trajectory was then used as the 2-D visual trace.

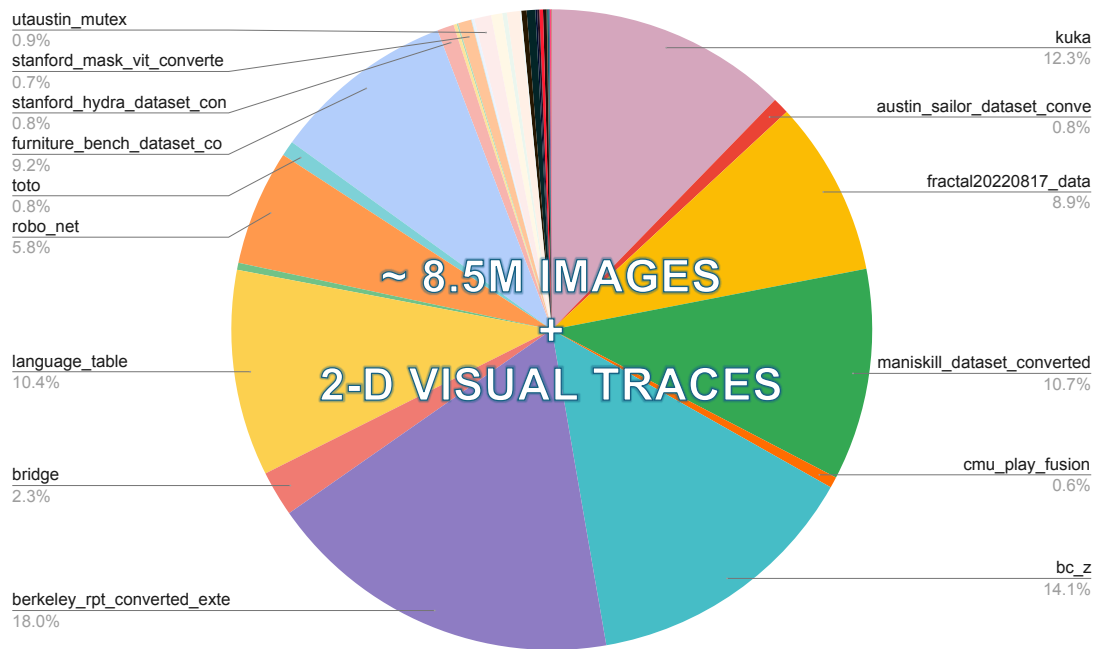


Figure C.2: Diagram of data distribution for LLARVA as shown in (4).

OXE Subset	Number of Image + 2-D visual trace pairs
kuka	1,044,466
austin_sailor_dataset_converted_externally_to_rlds	70,758
fractal20220817_data	753,647
maniskill_dataset_converted_externally_to_rlds	909,568
cmu_play_fusion	47,115
bc_z	1,198,963
berkeley_rpt_converted_externally_to_rlds_new	1,533,451
bridge	195,745
language_table	885,876
stanford_kuka_multimodal_dataset_converted_externally_to_rlds	30,128
robo_net	496,454
toto	65,527
furniture_bench_dataset_converted_externally_to_rlds	786,692
stanford_hydra_dataset_converted_externally_to_rlds	72,160
ucsd_pick_and_place_dataset_converted_externally_to_rlds	13,545
kaist_nonprehensile_converted_externally_to_rlds	6,512
stanford_mask_vit_converted_externally_to_rlds	57,012
utokyo_pr2_opening_fridge_converted_externally_to_rlds	2,276
berkeley_fanuc_manipulation	11,854
utaustin_mutex	72,461
taco_play	47,780
berkeley_autolab_ur5	19,621
austin_sirius_dataset_converted_externally_to_rlds	56,101
columbia_cairlab_pusht_real	5,486
stanford_robocook_converted_externally_to_rlds	22,894
roboturk	37,120
berkeley_cable_routing	7,797
nyu_franka_play_dataset_converted_externally_to_rlds	9,118
jaco_play	15,515
viola	15,146
tokyo_u_lsmo_converted_externally_to_rlds	2,398
austin_buds_dataset_converted_externally_to_rlds	6,771
dlr_sara_pour_converted_externally_to_rlds	2,695
utokyo_xarm_pick_and_place_converted_externally_to_rlds	1,381
utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds	6,545
dlr_edan_shared_control_converted_externally_to_rlds	746
dlr_sara_grid_clamp_converted_externally_to_rlds	1,543

Table C.1: More statistics about the LLARVA (4) dataset.

# Appendix D

## Extension of LISAt

In this appendix, we include several additional discussions:

- Appendix D.1 details the code release, including links to the codebases and datasets used in this project.
- Appendix D.2 outlines the prompt structure used for engineering the **GRES** dataset for LISAt, provides further details on its class distribution as well as its quality verification, and discusses additional experiments.
- Appendix D.3 presents additional details on the **PreGRES** dataset used to fine-tune LISAt<sub>PRE</sub>, discussing its composition and further evaluations.
- Appendix D.4 showcases qualitative results, highlighting both successful and failure cases, as well as instances where LISAt was penalized due to incomplete Ground Truth annotations generated by GeoSAM (GT).

### D.1 Code Release

The project page for this work is available [here](#). Our code for LISAt, derived from the Apache 2.0-licensed LISA codebase (127), as well as the curated datasets, are publicly released under the MIT license (or their respective licenses) and could also be found on the same page.

### D.2 More on GRES

#### Prompt Engineering

As outlined in Section 6.3, we used GPT-4o to generate the final prompt in two stages, detailed below.

### Prompt Engineering Stage 1

In the first stage, we input a  $512 \times 512$  chip into the model and prompt it, following the template below, to generate a sentence that accurately describes the item within the bounding box provided, as specified by the Ground Truth from xView

```
The size of the original image is (512,512).
This original image, where the image's origin is at the top left corner,
contains the following objects: {classes_list_str}.
Only focus on {class_name} in the image.
If {class_name} contains the word 'Other', remove the word 'Other' and
use only the second word in {class_name} describing the class. In
that case, make sure that second word in {class_name} starts with a
lowercase letter.
The following are the bounding boxes [x, y, width, height] of objects of
class {class_name}, where (x,y) represents the top left corner of
the bounding box, and 'width' represents the bounding box's width,
and 'height' represents the bounding box's height.
The bounding box of the {class_name} is at coordinates {bbox}.
Find visual features (color, shape, size, etc.) that can help find or
segment {class_name} in the image.
Generate a sentence (not a question) that can uniquely segment or
identify or find or locate {class_name} in this image, be concise
and clear.
```

Where `{classes_list_str}`, `{class_name}`, and `{bbox}` are the ground truth list of classes, the object class name or category, and the bounding box of the object from the xView dataset, bounding box and class annotations.

The model outputs a descriptive sentence in the variable `{unique_characteristics.query}`, which is then used to query the model again in the second stage, as shown below.

### Prompt Engineering stage 2

Once the uniquely descriptive sentence is generated, we asked the model using the template below to come with a question to which the given sentence in `{unique_characteristics.query}` will be the answer.

```

The size of the original image is (512,512).
Only focus on {class_name} in the image.
In the original image, where the image's origin is at the top left
corner, the object is a {class_name} located at bounding box
coordinates {bbox}.
The following are the bounding boxes [x, y, width, height] of objects of
class {class_name}, where (x,y) represents the top left corner of
the bounding box, and 'width' represents the bounding box's width,
and 'height' represents the bounding box's height:
This original image, where the image's origin is at the top left corner,
contains the following objects: {classes_list_str}.
If {class_name} contains the word 'Other', remove the word 'Other' and
use only the second word in {class_name} describing the class. In
that case, make sure that second word in {class_name} starts with a
lowercase letter.
{ ' located at bounding box coordinates {bbox}.' if include_bbox else
  '.'}
Please generate a query that would help locate this {class_name} in the
original image.
Your query will be the question to the answer provided by {
  unique_characteristics.query}.
For example, if the value contained in {unique_characteristics.query} is
  'Look for a long rectangular shape with distinct wheels, typically
  metallic or painted in color, connected to a truck cab at the front
  ', your query should be:
'Segment the blue car in the bottom right of the image with a long
rectangular shape with distinct wheels, typically metallic or
painted in color, connected to a truck cab at the front'
'Identify the blue car in the bottom right of the image with a long
rectangular shape with distinct wheels, typically metallic or
painted in color, connected to a truck cab at the front'
'Find the blue car in the bottom right of the image with a long
rectangular shape with distinct wheels, typically metallic or
painted in color, connected to a truck cab at the front'
'Locate the blue car in the bottom right of the image with a long
rectangular shape with distinct wheels, typically metallic or
painted in color, connected to a truck cab at the front'
>Show the blue car in the bottom right of the image with a long
rectangular shape with distinct wheels, typically metallic or
painted in color, connected to a truck cab at the front'.
Generate the query considering the sentence: {unique_characteristics.
query}
{ 'and the location described by the bounding box.' if include_bbox else
  '.'}
Make sure to vary the start of your queries with key words such as '
Segment, Find, Locate, Show, Identify' and similar synonyms. Do not
overuse one over the others.
Rephrase the generated query to make it sound better.
{ 'Do not mention or use any location-related info such as: top, near
the center in your query.' if not include_bbox else ''}
Do not output the exact bounding box coordinates, instead, output the
locations such as: bottom-left, top-right, top-left, bottom-right,
center, etc.
The response to the generated queries should be a JSON object in the
following format and contain nothing else:
The response to the generated query should be a sentence, not a question
.
Be concise and clear, start the sentence with: Locate, Segment, or
Identify.
{"query": "<your_query_here>"}

```



Where `{class_name}`, `{bbox}`, `{unique_characteristics.query}`, and `{class_name}` are the ground truth class name or category of the object class name or category, its bounding box and the unique characteristics obtained from GPT-4 (143) in the first stage.

The final query is then treated as the principal query. To enhance query diversity, we ask GPT to rephrase the principal query into two additional variants, resulting in three distinct queries per image. We then use GeoSAM to generate corresponding masks, forming image-queries-mask tuples.

## Dataset Quality Assurance

We use RGB images from the **xView** dataset (146), as referenced in this document. Although the dataset covers regions in South America, Africa, Europe, Asia, and Australia, we agree that LISAT and GRES would benefit from additional datasets from around the world, as shown in Figure 6.4.

For the classes of the targeted object referenced in the GRES natural language queries associated with each image, we inherited them from the **Quality Control and Gold Standards** method used in the **xView** paper (146). In their paper, the authors outline a three-tier quality assurance process: worker, supervisory, and expert stage. In the first stage, labelers reviewed each other’s annotations in a rotating manner. During the supervisory phase, the process included checks for duplicate or incorrect labels, geometry errors, incomplete annotation coverage, misaligned features, and empty image tiles. In the final expert stage, annotations were compared against a gold standard dataset. This reference dataset was developed by the paper’s co-authors and professional image analysts. It involves a manual labeling of six 1 km<sup>2</sup> image chips per batch. To meet the expert-level quality standard at this 3rd stage, annotation batches were required to achieve a minimum of 0.75 precision and 0.95 recall at a 0.5 Intersection over Union (IoU) threshold when evaluated against the gold standard.

For natural language queries in GRES, we generated three variations per RGB image and used cosine similarity to ensure they conveyed semantically equivalent information. Only those with a similarity/ alignment score of 0.9 or higher were retained.

Regarding ground truth segmentation masks produced via GeoSAM, we used only the cropped RGB regions defined by the bounding boxes in the **xView** dataset (146). Random batches of GeoSAM outputs were inspected by co-authors in a rotating manner. Their task was to validate or reject the generated masks. A randomly selected data point from a random batch was retained only if all participants unanimously agreed on its accuracy.

For the obtained data, we employ *Human Verification*, where multiple team members manually inspect randomly selected subsets of the dataset to verify the accuracy of the query-image-annotation triplets.

## GRES Dataset Summary

Table D.1 below shows the LISAT dataset distribution per class. We have also provided bar charts for the dataset distributions in Figures D.1 through D.5.

Object Category	Train (2.5k)	Train (4.5k)	Train (7.2k)	Val (0.5k)	Test (1.5k)	Test-L (0.5k)	Test-S (1k)
Truck w/Trailer Bed	142	298	469	25	100	34	66
Dump/Haul Truck	104	148	208	16	50	18	32
Bus	224	417	671	61	139	8	131
Facility	115	197	370	28	66	44	22
Car	247	546	914	65	182	2	180
Truck	240	518	932	75	173	12	161
Small Plane	6	18	39	2	7	3	4
Shed	80	152	249	7	51	16	35
Hut/Tent	26	46	82	9	16	8	8
Storage Tank	47	74	120	8	25	14	11
Truck w/Liquid Tank	21	29	45	3	10	4	6
Building	331	548	937	69	183	102	81
Helicopter	6	12	19	2	4	1	3
Passenger/Cargo Plane	107	135	198	11	45	25	20
Aircraft Hangar	25	39	73	6	13	9	4
Aircraft	3	15	29	0	5	3	2
Container Ship	31	72	102	5	24	11	13
Motor/Sail/Small Boat	32	58	87	7	20	2	18
Maritime Vessel	41	92	134	12	31	21	10
Crane Truck	33	48	70	2	16	5	11
Container Crane	12	25	38	4	9	0	9
Tower Crane	18	42	57	6	14	7	7
Engineering Vehicle	82	115	166	15	39	11	28
Excavator	84	115	161	12	39	10	29
Straddle Carrier	3	7	14	2	3	2	1
Passenger Vehicle	96	145	215	15	49	0	49
Pylon	104	140	177	6	47	34	13
Helipad	15	21	32	2	8	6	2
Loader/Dozer/Tractor	100	137	186	7	46	7	39
Damaged Building	61	151	226	8	51	37	14
Railway Vehicle	13	22	26	1	8	8	0
Locomotive	13	21	32	3	8	4	4
Tower Structure	16	30	41	1	11	6	5
Barge	17	42	59	5	14	13	1
Passenger Car	5	14	27	1	5	1	4
<b>Total</b>	2500	4489	7205	500	1500	488	1023

Table D.1: Summary of object categories across train, validation, and test sets

## Additional Experiments

### 1. GPT vs. Human vs. Template style Queries

We start by asking whether there is a difference in language style and complexity between queries generated by Large Language Models and real analysts. Though using GPT-generated queries is an effective strategy, it is important to consider the potential differences between the two.

To preliminarily investigate this, we conducted a small-scale comparative analysis using 10 test examples. We created two additional query variants: (1) human-like rewordings and (2) template-based queries referencing specific image regions. Our evaluation

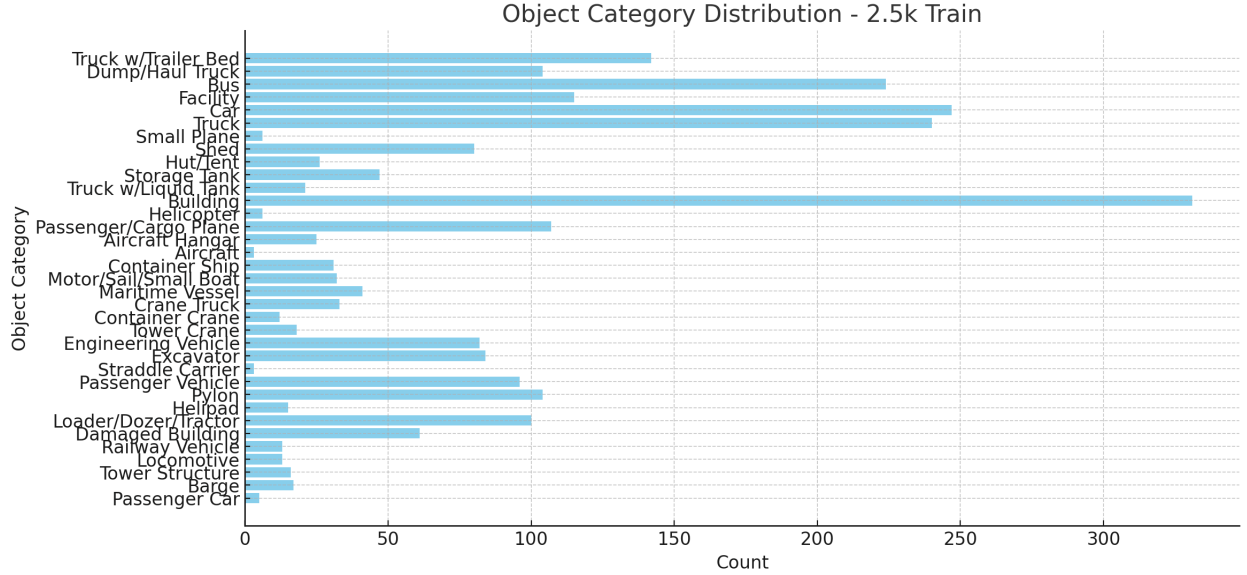


Figure D.1: Class distribution of 33% training set

showed in Table D.2 revealed that while GPT-style queries achieved slightly higher average performance on segmentation metrics, the differences were accompanied by relatively high variance, likely due to the tiny sample size.

This initial result suggests that GPT-generated queries are a reasonable proxy for human queries in the current setting, supporting the effectiveness of our dataset construction approach. However, we agree that a larger-scale collection of real human queries would provide a stronger validation and potentially improve the dataset further if augmented with such a collection.

Type of Queries (10 test examples)	cIoU ( $\pm$ )	gIoU ( $\pm$ )
LISAT on Template-style queries	$0.025 \pm 0.014$	$0.045 \pm 0.021$
LISAT on Human-entered queries	$0.037 \pm 0.022$	$0.063 \pm 0.040$
LISAT on GPT-style queries (GRES Data)	$0.050 \pm 0.036$	$0.099 \pm 0.045$

Table D.2: Performance comparison across different query types on 10 test examples.

## 2. LISAT vs. LISA on Natural Images

We also evaluated and reported in Table D.3 the performance of LISAT on the natural images test set from the LISA benchmark (127). **LISA-7B on LISA Natural Images Data** is the baseline model reported in the original paper. **LISA-7B (ft) on LISA**

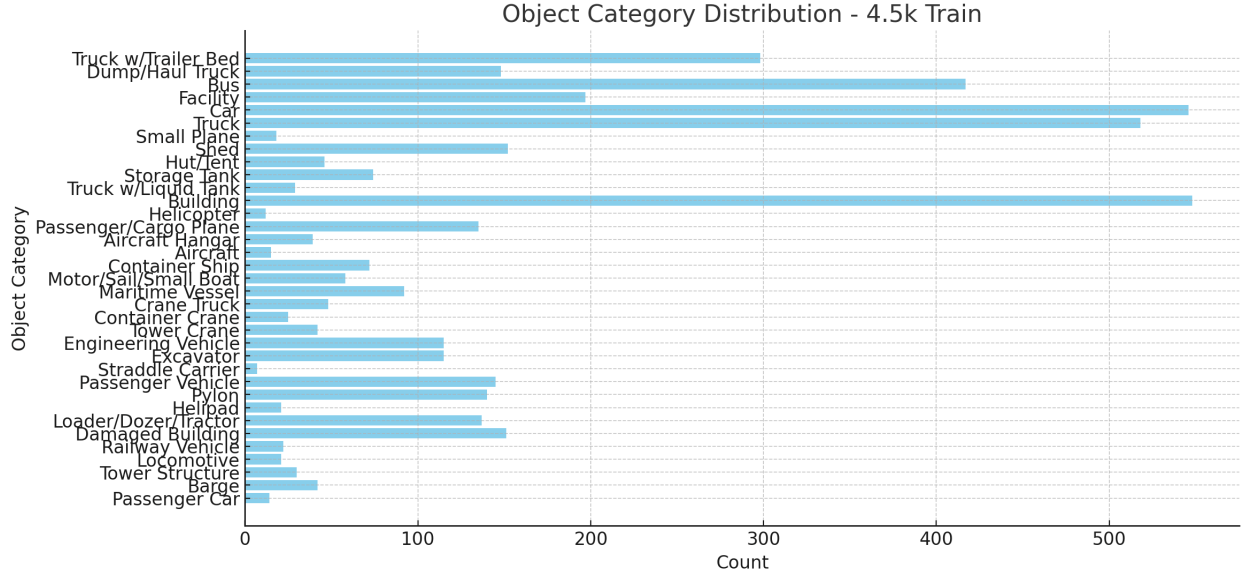


Figure D.2: Class distribution of 66% training set

**Natural Images Data** refers to the fine-tuned version, where (127) note that performance improves after fine-tuning on 239 complex-reasoning samples. **LISAT on LISA Natural Images Data** represents our LISAT model evaluated on the same test set, while **LISAT on GRES Data** shows its performance on the **GRES** dataset.

Type of Model and Data	cIoU	gIoU
LISA-7B on LISA Natural Images Data	0.341	0.368
LISA-7B (ft) on LISA Natural Images Data	0.484	0.473
LISAT on LISA Natural Images Data	0.326	0.341
LISAT (Ours) on GRES Data	0.245	0.275

Table D.3: Performance comparison across different models and datasets.

The results shown in Table D.3 indicate that LISAT does perform slightly worse than the original **LISA-7B** model and its fine-tuned version on this domain. Specifically, LISAT achieved a cIoU of 0.326 and gIoU of 0.341, compared to 0.341/0.368 for **LISA-7B** and 0.484/0.473 for the fine-tuned **LISA-7B (ft)**.

While LISAT is not optimized for natural image reasoning tasks, its performance is still in a comparable range to the baseline **LISA-7B** model. The difference is expected, as LISAT is designed for generalization across geospatial and abstract reasoning segmentation tasks, and has not been fine-tuned on the **LISA** dataset. Thus, while it does

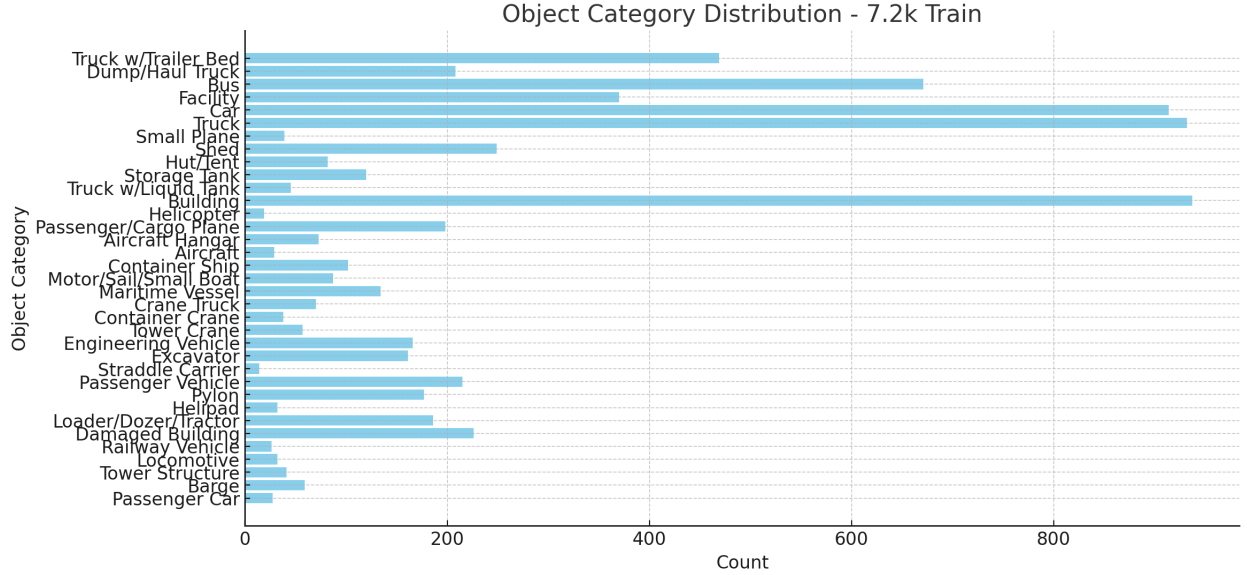


Figure D.3: Class distribution of 100% training set

not outperform models specialized or fine-tuned on natural image tasks, it remains competitive and shows promise as a more generalizable model.

### 3. LISat’s Latency Across Image Resolutions

Because assessing inference speed and computational requirements is important for evaluating practical deployment feasibility, we have included an analysis in Table D.4, which reports the average inference time per image-query pair on a single NVIDIA A100 GPU across different image resolutions.

Image Size	Object Size	cIoU ( $\pm$ )	gIoU ( $\pm$ )	Avg. Inference Time (s/image-query pair)
512 × 512	All	0.245 ± 0.023	0.275 ± 0.009	0.244
256 × 256	All	0.237 ± 0.029	0.207 ± 0.007	0.262
128 × 128	All	0.158 ± 0.019	0.130 ± 0.007	0.391
64 × 64	All	0.102 ± 0.010	0.061 ± 0.003	0.454
32 × 32	All	0.081 ± 0.007	0.042 ± 0.004	0.468

Table D.4: Effect of input image Size on performance and inference time (All object sizes). Measured on a single NVIDIA A100.

We found that at a standard resolution of 512×512, the model achieves 0.244 seconds per query, while maintaining competitive accuracy (cIoU: 0.245 ± 0.023, gIoU: 0.275 ± 0.009). As expected, inference becomes slower and less accurate at very low resolutions

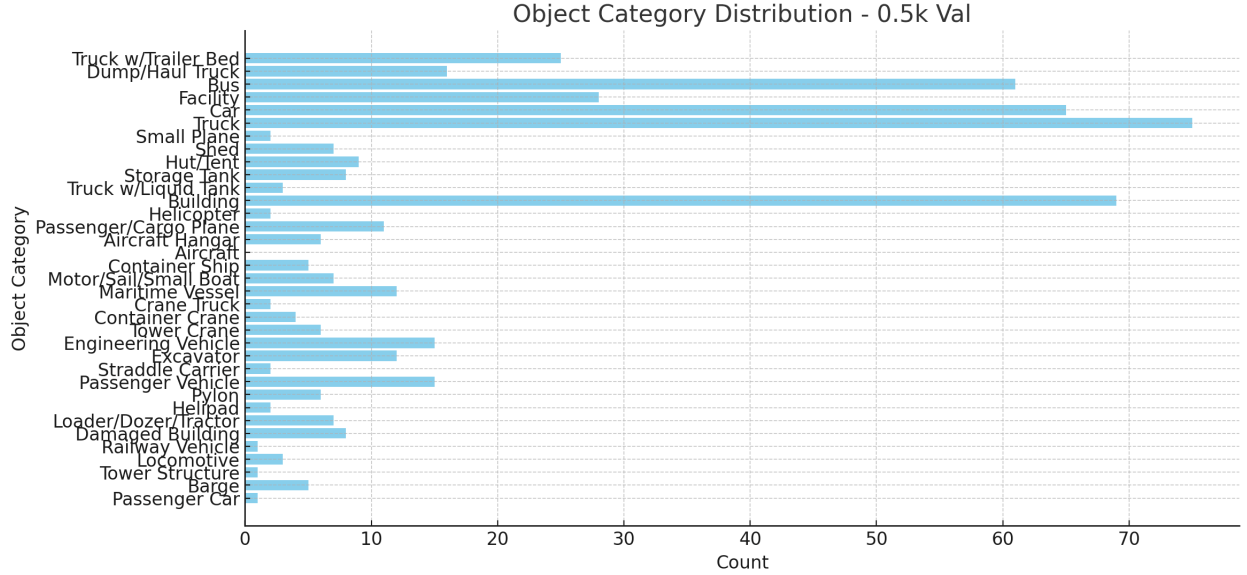


Figure D.4: Class distribution of validation set

(e.g.,  $32 \times 32$ ), where performance drops ( $\text{gIoU}$ :  $0.042 \pm 0.004$ ) and latency slightly increases (0.468s).

We believe the inference time increases at lower resolutions because the frozen vision encoder still processes inputs at a fixed size of  $512 \times 512$  in our case. This requires lower-resolution input images to be resized back to  $512 \times 512$ , which adds an overhead. These resized images also contain fewer details, which makes it harder for the model to perform well since our pipeline resizes all inputs to the fixed resolution required by the encoder before inference.

For very small images (e.g.,  $32 \times 32$ ), the additional overhead from resizing operations and suboptimal GPU utilization can slightly increase inference time, as shown in Table D.4.

This indicates that reducing input resolution significantly degrades visual quality without providing meaningful speed benefits, which supports the use of higher resolutions (e.g.,  $512 \times 512$ ) in deployment settings.

#### 4. LISAT<sub>pre</sub> vs. GPT-4o vs. GPT-o1

At the time of this work, we note that since GPT-4o (191) and GPT-o1 (192) do not explicitly output segmentation masks, they cannot be fairly compared with LISAT. A specialized prompt must be engineered to extract the coordinates of points along the contour lines for the target object. Instead, we compare them to LISAT<sub>PRE</sub>.

We found that GPT-4o (191) and GPT-o1 (192) yielded identical scores across all metrics and benchmark datasets, while significantly underperforming compared to

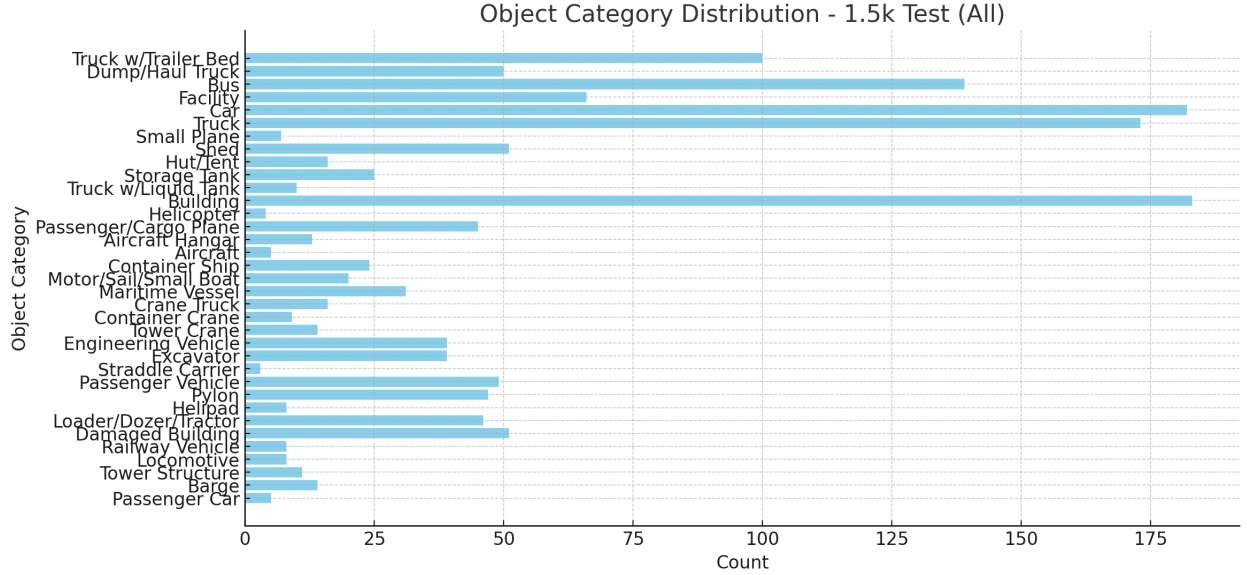


Figure D.5: Class distribution of testing set

LISAT<sub>PRE</sub> on the PreGRES test data. We verified that this results from both models returning generic or irrelevant outputs (e.g., hallucinated captions, answers unrelated to the query, or blank responses), likely due to their lack of grounding in geospatial semantics and structured output generation.

While these models represent the state of the art in general-purpose multimodal reasoning, they often require carefully crafted prompts to perform meaningfully on domain-specific tasks such as geospatial captioning or other domain-related VQA. This highlights the need for specialized VLMs like LISAT, which natively support geospatial semantics and reasoning. We refer the interested reader to (124) for more details.

## 5. More on Future Work

Building on the promising performance of LISAT, we outline several directions for future work to enhance both the model and the GRES dataset:

### a) Incorporation of Temporal Data

It will be interesting to extend LISAT’s capabilities by incorporating temporal geospatial data, enabling the model to reason over frame sequences. This includes investigating the effects of frame-rate downsampling and adapting architectures that leverage temporal vision encoders such as TimeSformer (193) and ViViT (194), as well as contrastive video-text pretraining approaches like VideoCLIP (195), in combination with different language encoders. These explorations aim to identify the most effective architectural combinations for spatiotemporal grounding and reasoning in remote sensing contexts.

b) **Expansion to Additional Modalities**

To enhance generalization and robustness, it will be interesting to integrate additional modalities such as synthetic aperture radar (SAR), LiDAR, aerial imagery, and elevation data (e.g., digital surface models, DSM), in both static and temporal settings. Once collected and processed, these modalities will broaden the applicability of the model and enable it to handle more dynamic and realistic geospatial scenarios. Incorporating these diverse inputs will also help evaluate LISAT’s cross-modal generalization capabilities.

c) **Dataset Enrichment and Potential Bias Mitigation**

For future iterations of the GRES dataset, it will be valuable to augment it with additional publicly available datasets such as SpaceNet (196) and COWC (197). This enrichment will help address existing dataset potential biases and improve the robustness and fairness of LISAT across a wider range of environmental and sensor conditions.

d) **Efficient Model Variants for Deployment**

For deployment in resource-constrained environments, future avenues include exploring model compression techniques such as knowledge distillation, quantization, and pruning. These approaches will enable us to reduce model size and improve inference efficiency while maintaining competitive performance, thereby supporting broader accessibility and real-time applications of LISAT.

Through these efforts, we believe LISAT will turn into an even more comprehensive and generalizable foundation model for geospatial-language understanding, capable of reasoning across modalities and time with increased accuracy and efficiency.

## D.3 More on PreGRES

We conducted additional evaluations of  $\text{LISAT}_{\text{PRE}}$ . We show evaluation results on the NWPU Caption in Table 6.5, RSICD in Table D.6, and Sidney-Caption in Table D.7. We also ran Count, Presence, Comparison and Area evaluation as was done in (129) in Table D.9.

## D.4 Qualitative Analysis

In this section, we present a qualitative analysis of the model’s performance, showcasing a range of success cases section D.4, failure cases section D.4, and instances where the ground truth (GT) was erroneous section D.4. Success cases shown in Table D.10, Table D.11, Table D.12, Table D.13, and Table D.14 highlight scenarios where the model successfully aligns with the expected outcomes, demonstrating its ability to handle complex tasks accurately. Failure cases shown in Table D.15, however, indicate situations where the model struggles due to challenges such as occlusion, poor lighting, or ambiguous object representations,



Task	Data Source	Train Images	Train QA Pairs	Test Images	Test QA Pairs
Image Captioning	NWPU-Captions	25200	125894	3150	1093
	RSICD	8734	17813	1093	1093
	RSITMD	4291	20096	-	-
	Sydney-Captions	497	2294	58	58
	UCM-Captions	1680	7999	210	210
Visual Question Answering	RSVQA-LR	572	57223	100	10004
	RSVQA-HR	6251	625340	2226	222684
	FloodNet	1448	4511	-	-
	RSIVQA	5401	19218	-	-
Visual Grounding	DIOR-RSVG	9466	19643	7936	18677
Region-level Captioning	DIOR-RSVG	9466	19643	-	-
Scene Classification	NWPU-RESISC45	31500	31500	-	-
<b>Total</b>	-	<b>104506</b>	<b>951174</b>	<b>14773</b>	<b>253819</b>

Table D.5: Overview of data sources and statistics

Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
VLAD + RNN (148)	49.38	30.91	22.09	16.77	19.96	42.42	103.92
VLAD + LSTM (148)	50.04	31.95	23.19	17.78	20.46	43.34	118.01
mRNN (147)	45.58	28.25	18.09	12.13	15.69	31.26	19.15
mLSTM (147)	50.57	32.42	23.29	17.46	17.84	35.02	31.61
mGRU (198)	42.56	29.99	22.91	17.98	19.41	37.97	124.82
mGRU embedword (198)	60.94	46.24	36.80	29.81	26.14	48.20	<b>159.54</b>
CSMLF (106)	57.59	38.59	28.32	22.17	21.28	44.55	52.97
SAA (163)	59.35	45.11	35.29	28.08	26.11	49.57	132.35
Soft-attention (199)	65.13	49.04	39.00	32.30	26.39	49.69	90.58
SD-RSIC (176)	64.50	47.10	36.40	29.40	24.90	51.90	77.50
RTRMN (semantic) (177)	62.01	46.23	36.44	29.71	28.29	<b>55.39</b>	151.46
RTRMN (statistical) (177)	61.02	45.14	35.35	28.59	27.51	54.52	148.20
SVM-D BOW (178)	61.12	42.77	31.53	24.11	23.03	45.88	68.25
SVM-D CONC (178)	59.99	43.47	33.55	26.89	22.99	45.57	68.54
MLAT (200)	66.90	51.13	41.14	34.21	27.31	50.57	94.27
Post-processing (179)	62.90	45.99	35.68	28.68	25.30	47.34	75.56
RS-GPT4V (129)	70.32	54.23	<b>44.02</b>	<b>36.83</b>	30.10	53.34	102.94
LLaVA-v1.5-7b (142)	38.36	18.27	8.46	3.57	14.64	27.36	16.96
LLaVA-v1.6-7b (141)	29.31	13.40	6.00	2.44	13.11	24.40	10.69
LISAT <sub>PRE</sub> (Ours)	<b>72.51</b>	<b>54.98</b>	43.77	36.10	<b>30.28</b>	53.80	118.39

Table D.6: Comparison of various models for LISAT<sub>PRE</sub> on RSICD

leading to incorrect predictions or missed detections. These cases reveal areas where model improvements are needed, particularly in dynamic environments or with less structured input data. Finally, GT mistake cases, as shown in Table D.16 refer to instances where the GT was erroneous but the model aligns with the expected ground truth annotations. The model is penalized here due to inherent inconsistencies in the dataset from the mask labeling with GeoSAM. These cases reveal the challenges posed by noisy or ambiguous ground truth data, highlighting the importance of dataset refinement and improved model calibration to reduce such errors. Together, these cases provide valuable insights into the model’s performance,

Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE L	CIDEr
VLAD + RNN (148)	56.58	45.14	38.07	32.79	26.72	52.71	93.72
VLAD + LSTM (148)	49.13	34.12	27.60	23.14	19.30	42.01	91.64
mRNN (147)	51.30	37.50	20.40	19.30	18.50	-	161.00
mLSTM(147)	54.60	39.50	22.30	21.20	20.50	-	186.00
mGRU (198)	69.64	60.92	52.39	44.21	31.12	59.17	171.55
mGRU embedword (198)	68.85	60.03	51.81	44.29	30.36	57.47	168.94
CSMLF (106)	59.98	45.83	38.69	34.33	24.75	50.18	75.55
SAA (163)	68.82	60.73	52.94	45.39	30.49	58.20	170.52
Soft-attention (199)	73.22	66.74	62.23	58.20	39.42	71.27	249.93
Hard-attention (199)	75.91	66.10	58.89	52.58	38.98	71.89	218.19
SD-RSIC (176)	72.40	62.10	53.20	45.10	34.20	63.60	139.50
SVM-D BOW (178)	77.87	68.35	60.23	53.05	37.97	69.92	227.22
SVM-D CONC (178)	75.47	67.11	59.70	53.08	36.43	67.46	222.22
Post-processing (179)	78.37	69.85	63.22	57.17	39.49	71.06	255.53
LLaVA-v1.5-7b (142)	41.04	19.62	10.80	4.69	13.71	31.38	10.89
LLaVA-v1.6-7b (141)	32.25	17.15	9.98	5.92	14.11	29.17	12.20
RS-GPT4V (129)	<b>82.26</b>	<b>75.28</b>	<b>68.57</b>	<b>62.23</b>	<b>41.37</b>	<b>74.77</b>	<b>273.08</b>
LISAT <sub>PRE</sub> (Ours)	77.92	68.30	60.75	54.24	38.50	69.92	216.36

Table D.7: Comparison of various models for LISAT<sub>PRE</sub> on Sydney-Captions

guiding future research and optimizations.

## Success Cases of LISAt

In this subsection, we present a selection of successful cases where LISAT accurately predicted object categories and configurations. These examples highlight the model’s ability to generalize and perform well under varied conditions, demonstrating its effectiveness in real-world applications.

## Failure Cases of LISAt

We examined failure cases where LISAT struggled to make accurate predictions in section 6.5. Some of these instances, where the model’s performance could be improved, highlight the challenges it faces under complex conditions, such as cloudy or ambiguous scenes as shown in Table D.15.

## Ground Truth Error Cases

Table D.16 displays cases where the model’s predictions are affected by errors in the ground truth data. These errors highlight discrepancies between the model’s output and the labeled data, shedding light on limitations within the dataset and the potential impact on evaluation metrics.

Vision Encoder	Language Encoder	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE L	CIDEr	SPICE
UCM-Captions									
CLIP	Llama 2	85.57	79.02	73.81	69.03	45.49	80.10	328.82	52.21
CLIP336	Llama 2	84.86	77.81	72.06	66.97	44.70	78.97	324.61	50.46
SAT-CLIP	Llama 2	41.24	32.38	13.90	8.82	28.30	30.41	8.15	8.15
Geo-CLIP	Llama 2	44.57	26.22	17.37	12.77	15.61	32.22	44.64	11.67
RemoteCLIP	Llama 2	85.95	79.00	73.38	68.31	45.80	79.99	330.94	52.17
CLIP	Vicuna	84.93	77.80	72.05	66.68	45.62	80.04	329.32	52.00
CLIP336	Vicuna	85.40	78.81	73.34	68.28	45.84	79.66	324.89	51.55
SAT-CLIP	Vicuna	47.21	29.55	21.21	16.87	17.35	34.20	63.92	15.08
Geo-CLIP	Vicuna	53.77	37.86	29.50	24.56	21.77	42.86	109.20	21.15
RemoteCLIP	Vicuna	<b>88.23</b>	<b>82.07</b>	<b>77.08</b>	<b>72.34</b>	<b>47.78</b>	<b>83.13</b>	<b>355.32</b>	<b>54.15</b>
NWPU-Captions									
CLIP	Llama 2	87.25	77.53	69.89	63.53	43.33	76.59	180.81	31.38
CLIP336	Llama 2	86.70	76.38	68.43	62.04	42.55	75.56	176.54	30.75
SAT-CLIP	Llama 2	69.51	50.90	39.28	31.48	25.83	52.34	68.94	16.15
Geo-CLIP	Llama 2	74.36	58.26	47.59	39.96	30.12	58.48	97.67	19.78
RemoteCLIP	Llama 2	87.25	77.67	70.06	63.76	43.44	76.64	181.01	31.48
CLIP	Vicuna	86.62	76.76	69.03	62.59	43.09	76.18	179.96	31.09
CLIP336	Vicuna	87.47	77.79	70.13	63.78	43.47	76.42	181.94	31.15
SAT-CLIP	Vicuna	75.87	60.51	50.35	43.00	31.78	60.51	105.53	21.25
Geo-CLIP	Vicuna	77.98	63.94	54.24	46.96	34.10	63.87	121.54	23.44
RemoteCLIP	Vicuna	<b>88.5</b>	<b>79.3</b>	<b>72.0</b>	<b>65.8</b>	<b>44.4</b>	<b>77.5</b>	<b>185.7</b>	<b>32.2</b>
RSICD									
CLIP	Llama 2	60.51	43.02	32.34	25.60	25.41	46.20	76.30	25.46
CLIP336	Llama 2	70.05	51.72	40.06	32.36	28.19	50.45	107.18	28.35
Geo-CLIP	Llama 2	45.86	21.97	11.90	7.16	15.73	22.60	22.60	12.88
RemoteCLIP	Llama 2	68.19	49.65	38.25	30.74	27.64	49.91	101.05	27.94
SAT-CLIP	Llama 2	43.68	19.17	9.82	5.60	14.61	26.65	16.71	10.95
CLIP	Vicuna	62.46	44.31	33.36	26.47	25.65	47.08	81.22	25.56
CLIP336	Vicuna	70.40	52.09	40.44	32.79	28.50	50.74	108.37	28.82
SAT-CLIP	Vicuna	45.95	22.68	13.00	8.23	16.02	29.02	24.77	12.76
Geo-CLIP	Vicuna	49.19	26.41	16.17	10.80	17.67	31.74	31.44	15.14
RemoteCLIP	Vicuna	<b>72.51</b>	<b>54.98</b>	<b>43.77</b>	<b>36.10</b>	<b>30.28</b>	<b>53.80</b>	<b>118.39</b>	<b>30.54</b>
Sydney-Captions									
CLIP	Llama 2	78.59	69.78	62.50	56.35	39.09	70.00	220.50	45.20
CLIP336	Llama 2	78.48	69.35	62.25	56.14	38.54	68.98	211.95	43.25
SAT-CLIP	Llama 2	58.40	45.47	37.93	32.27	25.75	46.95	85.67	22.96
Geo-CLIP	Llama 2	68.91	56.44	48.91	43.09	30.75	55.86	153.84	31.33
RemoteCLIP	Llama 2	76.19	66.31	58.57	52.27	37.30	68.12	201.64	43.99
CLIP	Vicuna	76.42	67.79	60.49	53.95	38.02	68.78	204.76	44.07
CLIP336	Vicuna	77.68	<b>68.70</b>	61.37	<b>55.35</b>	38.41	<b>70.37</b>	213.92	45.22
SAT-CLIP	Vicuna	69.58	58.86	51.80	46.45	32.24	58.43	182.93	34.65
Geo-CLIP	Vicuna	71.68	60.83	53.67	47.76	34.33	61.49	177.19	36.60
RemoteCLIP	Vicuna	<b>77.92</b>	68.30	<b>60.75</b>	54.24	<b>38.50</b>	69.92	<b>216.36</b>	44.04

Table D.8: Comparison of vision and language encoders for LISAT<sub>PRE</sub> on UCM-Captions, NWPU-Captions, RSICD, and Sydney-Captions

Model	Count	Presence	Comparison	Area
RSVQA (163)	67.01	87.46	81.50	85.24
EasyToHard (181)	69.22	90.66	87.49	85.92
Bi-Modal (182)	<b>72.22</b>	91.06	91.16	86.27
SHRNet (183)	73.87	91.03	90.48	<b>86.35</b>
LLaVA-1.5 (142)	26.81	54.72	66.22	1.45
InternLM-XC2 (184)	26.91	55.74	64.89	5.94
RS-GPT4V (129)	-	91.17	91.70	-
GeoChat (185)	-	91.09	90.33	-
Full-FT (129)	70.48	91.10	92.23	86.00
LoRA (129)	70.34	92.24	92.10	85.84
MoE LoRA (129)	71.06	91.10	<b>92.55</b>	85.82
LLaVA-v1.5-7b (142)	18.66	53.98	66.22	58.00
LLaVA-v1.6-7b (141)	19.65	57.53	62.32	62.00
LISAT <sub>PRE</sub> (Ours)	70.24	<b>92.36</b>	92.20	61.43

Table D.9: Performance metrics for LISAT<sub>PRE</sub> on the RSVQA\_LR






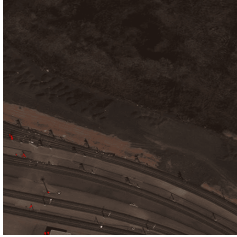
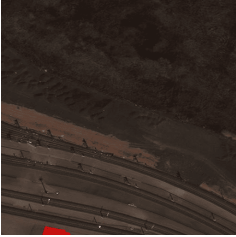
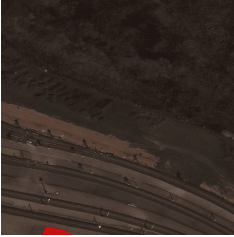
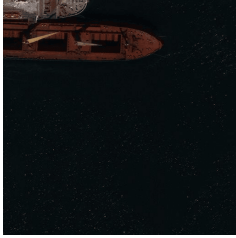
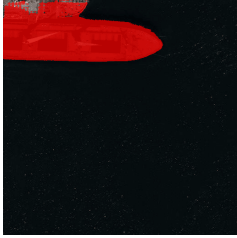





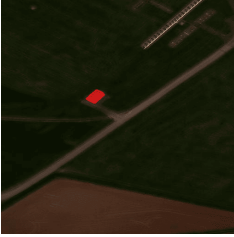








Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Identify the excavator by locating the bright yellow arm and bucket against the darker background.				
Locate the building with a beige facade and a dark brown roof in the image.				
Locate the large, elongated structure with stacked rectangular containers and a reddish-brown deck, characteristic of a container ship, against the dark water background.				
Locate the building in the center-left of the image.				
Locate the long, green vehicle with rectangular windows and wheels, positioned horizontally across the image.				
Locate the building in the top-left of the image.				

Table D.10: Comparison of predictions and ground truth across models



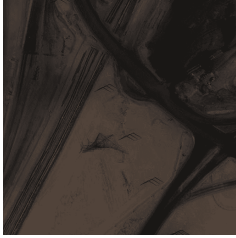











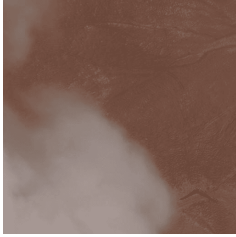

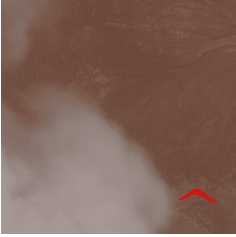
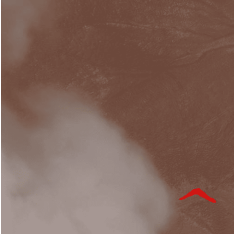
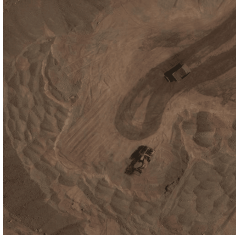
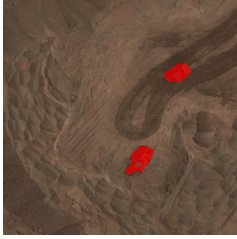
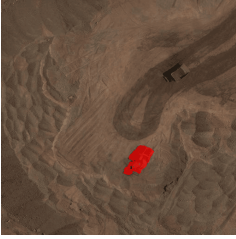
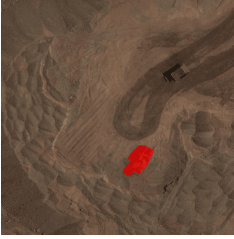




Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Identify the triangular metal structure with intersecting lines, standing vertically in the image.				
Identify the circular structure with a metallic appearance and distinct shadow, contrasting against the surrounding terrain.				
Identify the pylon in the top-left area of the image.				
Identify the pylon located in the bottom-right of the image.				
Identify the engineering vehicle with a metallic appearance and distinct geometric shapes against the brown background.				
Identify the damaged building with an irregular, fragmented roof structure and scattered debris contrasting with surrounding vegetation.				

Table D.11: Comparison of predictions and ground truth across models (Cont.)

Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Segment the damaged building located in the top-right of the image.				
Identify the building in the center-left of the image.				
Identify the building with a unique vertical dark brown structure with a slight curvature on the edge.				
Identify the large, rectangular building with a dark roof and multiple visible roof fixtures.				
Locate the trailer bed in the top-right of the image, characterized by a long rectangular shape with distinct wheels, typically metallic or painted in color, attached to a truck cab.				
Identify the liquid tank in the top-right of the image with a long rectangular shape connected to a truck cab at the front.				

Table D.12: Comparison of predictions and ground truth across models (Cont.)



Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Locate the building with a reddish-brown roof next to a dark black structure in the image.				
Identify the damaged building in the center of the image.				
Locate the maritime vessel in the bottom-right of the image.				
Identify the building with a rectangular shape, dark roof, and noticeable white lines across its surface, set against a brownish background with green areas nearby.				
Identify the building with a grayish roof and white linear features.				
Locate the engineering vehicle in the top-left of the image.				

Table D.13: Comparison of predictions and ground truth across models (Cont.)



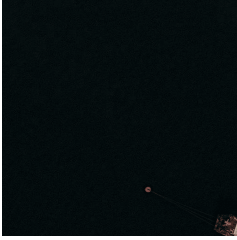
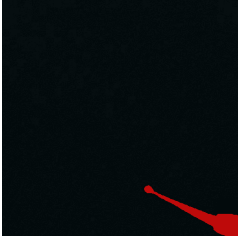
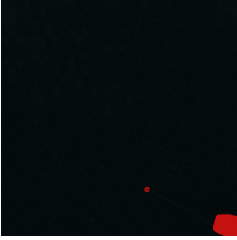
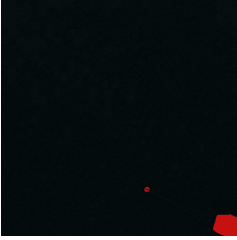
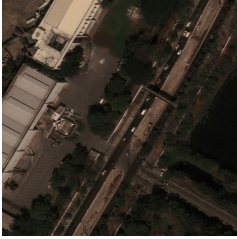
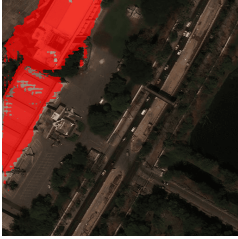
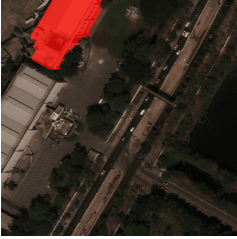
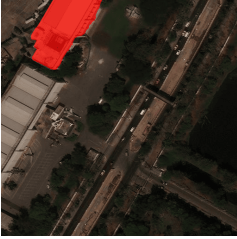
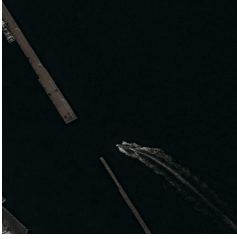
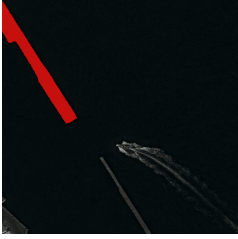
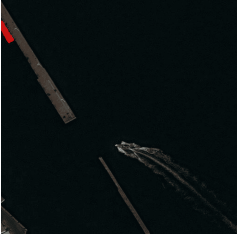
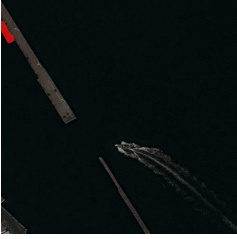





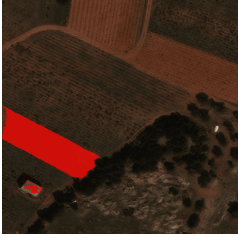



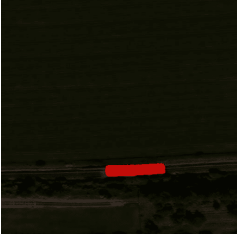
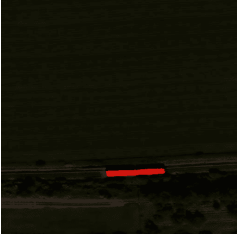
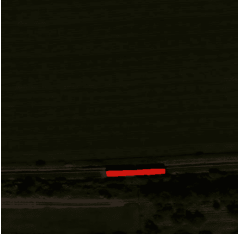
Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Locate the large rectangular structure with stacked, multicolored containers floating on water as the container ship.				
Locate the building in the top-left corner of the image.				
Identify the maritime vessel near the top-left corner of the image.				
Identify the aircraft hangar with the large rectangular structure and curved roof, displaying a uniform beige coloration and surrounded by open areas.				
Identify the large rectangular brown building with a flat roof surrounded by vegetation.				
Identify the railway vehicle with an elongated, rectangular shape and a metallic texture contrasting against the dark background.				

Table D.14: Comparison of predictions and ground truth across models (Cont.)






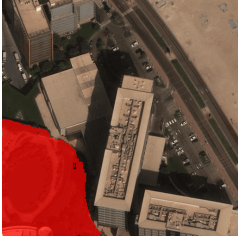




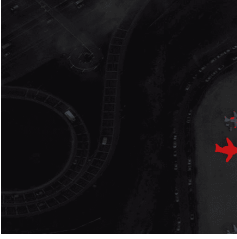
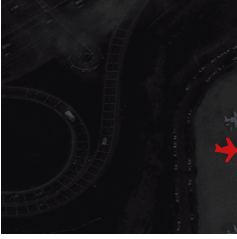









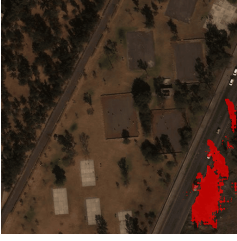


Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Locate the facility in the top-center of the image for identification.				
Find the facility in the bottom-left corner of the image.				
Identify the plane in the bottom-right of the image.				
Locate the barge in the top-left of the image.				
Locate the building with a distinctive light gray color and rectangular shape against the darker background.				
Identify the trailer in the bottom-right of the image with a distinct shape, typically metallic or painted, connected to a truck cab at the front.				

Table D.15: Failure cases



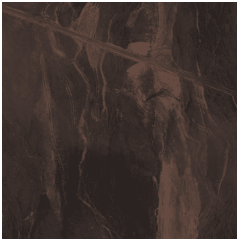
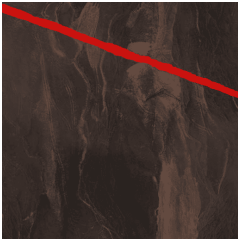
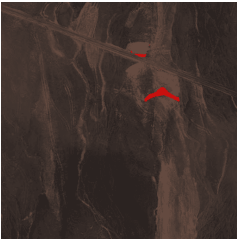
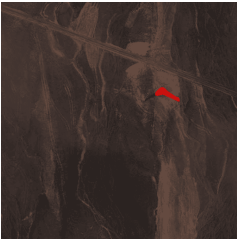
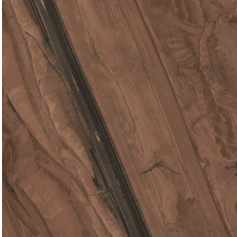
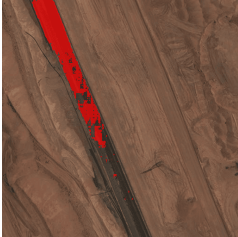


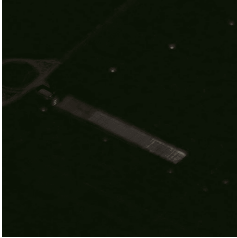
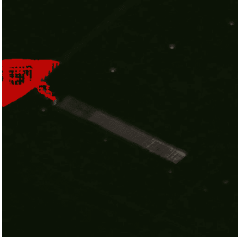
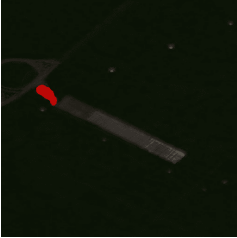
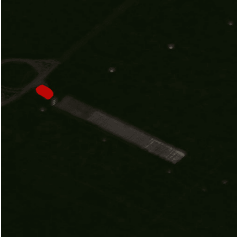
Queries	RGB	LISA	LISAt (Ours)	Ground Truth
Identify the pylon in the top-right area of the image.				
Identify the vertical, metallic structure with a lattice framework contrasting against the brown, earthy background.				
Identify the building with a large, rectangular structure and a distinct reddish-brown roof, surrounded by greenery.				

Table D.16: GT Mistake cases