5-2024

Enabling and optimizing multi-modal sense-making for human-AI interaction tasks

Dulanga Kaveesha Weerakoon WEERAKOON MUDIYANSELAGE
Singapore Management University, mweerakoon.2019@phdcs.smu.edu.sg

ENABLING AND OPTIMIZING MULTI-MODAL SENSE-MAKING FOR HUMAN-AI INTERACTION TASKS

WEERAKOON MUDIYANSELAGE DULANGA KAVEESHA WEERAKOON

Singapore Management University

2024

Enabling and Optimizing Multi-Modal Sense-Making for Human-AI Interaction Tasks

Weerakoon Mudiyanselage Dulanga Kaveesha Weerakoon

Submitted to School of Computing and Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Archan MISRA (Supervisor / Chair)
Vice Provost (Research); Lee Kong Chian Professor of Computer Science
Singapore Management University

Vigneshwaran SUBBARAJU (External Co-Supervisor)
Senior Research Scientist
Agency for Science Technology and Research (A*STAR)

Kotaro HARA
Assistant Professor of Computer Science
Singapore Management University

Jing JIANG
Professor of Computer Science
Singapore Management University

Nairan ZHANG (External Examiner)
Engineering Manager
Amazon

Singapore Management University
2024

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.

Weerakoon Mudiyanselage Dulanga Kaveesha Weerakoon
17 May 2024

Abstract

The rapid pace of adoption of mixed-reality in tandem with advances in NLP and computer vision have opened up unprecedented opportunities for more naturalistic interaction interfaces which underpin Human-AI collaborative applications such as spatial computing and interactive conversational agents. One notable example is the emergence of interactive virtual assistants, which facilitate more natural communication of instructions and queries through modalities like voice and text. This trend is driving the development of innovative ubiquitous, mixed-reality computing applications. Such interactive, natural communication is also critical to support advances in human-robot interactive co-working, across a variety of industrial, commercial and home environments. Conventional voice-based conversational agents, exemplified by technologies such as Apple's Siri and Amazon's Alexa, are evolving into increasingly multi-modal systems, which can now support the comprehension of human instructions through a combination of language, gestures, and visual inputs. The intelligence behind these conversational agents relies on sophisticated Deep Neural Network (DNN) models (e.g., [57, 43, 96]). Sophisticated Deep Neural Network (DNN) based architectures (e.g., Transformers [100]), which underlie the recent emergence of Large Language Models (LLMs) and Vision Language Models (VLMs), have recently dramatically enhanced the ability of AI software to comprehend a mix of visual and natural textual/verbal cues. While these models exhibit increasing accuracy, their computationally intensive nature and large model sizes pose challenges for supporting low-latency, on-device execution of inference tasks, especially on resource-constrained wearable and Internet of Things (IoT) devices like Microsoft HoloLens or Nvidia Jetson platforms. Thus, my research is centred on enabling the execution of these multi-modal human interactive

tasks, with a specific focus on comprehending human visual grounding instructions [121, 129], on resource-constrained devices. The goal is to achieve low-power, low-latency execution while maintaining comparable task accuracy, thereby preserving interactivity.

Natural human-human interaction is inherently multi-modal, as we use a variety of modalities including verbal commands, gestures and facial expressions, visual cues, gaze and even vocal nuances (e.g., tone and rhythm) to mutually convey our intent. Motivated by such human-human interaction scenarios, this thesis broadly investigates some methods to enable multi-modal sense-making for human issued instructions or queries in resource-constrained wearable and edge devices. In particular, we consider *object acquisition* as an exemplary task for human-AI collaboration that can benefit from enabling the support for comprehending naturalistic multi-modal instructions. To address this, we leverage Referring Expression Comprehension (REC) or Visual Grounding models [121, 129] developed in computer vision and NLP literature. These models, when provided with an image along with verbal and/or gestural inputs, identify the bounding box of the referred object. We then introduce a number of sense-making models and optimization techniques to support low-latency execution of such models for inferencing on pervasive devices.

In this thesis, our emphasis will be predominantly on exploring diverse dynamic optimizations for the comprehension of task instructions. Throughout these investigations, we rely on a common guiding principle which underscores our approach: the acknowledgement that ***not all instructions pose the same level of task complexity***. To illustrate, consider the varying complexities introduced by different types of instructions. In a cluttered environment, identifying a target object often necessitates a more intricate execution pipeline to ensure accurate identification. Users may employ a combination of language instructions and pointing gestures, which can aid the model in disambiguating among closely situated objects. Consequently, the presence of multiple modalities helps alleviate task complexity. Conversely, in a less cluttered space, a simple pointing gesture may suffice for object identification, requiring a less complex execution pipeline. This nuanced understanding of task complexities serves as the foundation for the dynamic optimization techniques explored in subsequent chapters.

This dissertation is organized into two parts. *Part I: Image-based Human Instruction Comprehension* focuses on studying model optimizations applied to REC models, which process a single static image along with language and, optionally, gestural modalities. In *Part II: Video-based Human Instruction Comprehension*, we extend our methodologies to more complex scenarios involving videos as vision input, moving beyond single static images.

We initially substantiate the significance of pointing gestures in enhancing task accuracy for object acquisition tasks. To demonstrate this, we present M2Gestic [106] system that combines neural-based text parsing with a novel knowledge-graph traversal mechanism, over a multi-modal input of vision, natural language text and pointing for achieving close-to-human task accuracy.

We then apply the paradigm of dynamic optimizations into vision and language based REC models. To this end, we develop SoftSkip [108], a multi-modal pruning strategy for REC tasks. SoftSkip uses language features to dynamically skip computations in a REC model. Key to this approach is a paradigm termed *soft-skipping* where a computational block that is decided to be skipped by the pruning model is approximated with a light-weight computation.

We further introduce the dynamic optimization paradigm for REC tasks which fuses vision, language, and pointing gestures to jointly reduce processing energy overheads and latency on a pervasive device. To tackle this, we introduce COSM2IC [107]; key to this approach is to predict the task complexity using a lightweight Task Complexity Predictor (TCOP) thereby invoking an adequate execution pipeline for the predicted complexity.

With the proposed Commit-And-Switch (CAS) paradigm, our focus shifts from the singular goal of reducing processing energy overheads, as seen in SoftSkip and COSM2IC, to a more holistic approach of reducing both sensing and processing energy overheads. In CAS, we first commit to a computational pipeline involving low overheads and a subset of available sensors. The task context estimated by this pipeline is then used to optionally switch to another energy-intensive DNN pipeline and activate additional sensors.

Until now, our proposed methodologies have been applied to REC models that operate on single static images. However, not all task instructions can be adequately characterized by a single static image. Some instructions may pertain to a moving object, an action performed by another user, or involve the user pointing to multiple objects within a single instruction. In part II, to address these dynamic scenarios, we extend our approach to Video Referring Expression Comprehension (VREC) models, optimizing them through our dynamic optimization paradigm.

With the aim of integrating dynamic model optimizations into VREC models, we introduce two approaches named GRefExSel and NeuroViG. Central to both methods is the selection of a set of *anchor-frames* from the raw video to execute a complex VREC pipeline efficiently. NeuroViG refines GRefExSel by integrating an Event camera with substantially lower sensing overhead. The Event camera acts as a trigger sensor, selectively activating the RGB camera only for the chosen anchor-frames, thereby reducing both processing and sensing overheads.

Acknowledgements

Embarking on a PhD journey is indeed a long and challenging endeavour. The successful completion of this journey would not have been possible without the invaluable assistance and guidance received from many.

First and foremost, I extend my sincere gratitude to my advisor, Prof. Archan Misra, for his immense support and counsel. It was sheer luck that led me to find your profile when I applied for an internship as an undergraduate who had no clue about research and no intention of pursuing a PhD at that point. I am so grateful for you taking a chance on me to hire me as an intern. I am indebted for your continued faith in me, from offering me a Research Engineer position to accepting me as your PhD student. You were my inspiration and motivation to embark on this arduous journey. I am blessed to have been advised by you and could not have had a better advisor. I can not thank you enough for all of the support you have given me, for attending to all my requests and for being so compassionate and understanding even when I lost my enthusiasm due to continuous paper rejections. You saw potential in me even when I doubted myself and failed to achieve remarkable outcomes. However, you never miss a subtle wake-up call of 'Average Performance' when I stumble into the comfort of complacency. Your constant will to improve, kindness, passion for work, genuine support and effort in building a safe, fun-filled environment to cope with personal and emotional challenges out on foreign soil were inspirational. Your dedication to ensuring that PhD students maintain good mental health is admirable. I am so grateful for all of the opportunities that you have given me and continue to do so when I start my professional career. All that I've accomplished, the person I am now, and my future professional growth I, attribute to you. You are truly exceptional and the best advisor a student could have asked for!

I also want to express my sincere appreciation to Dr Vignesh, who served as a co-advisor throughout my PhD journey. I first met you when I joined SMU as an intern, and you mentored me during my undergraduate internship. Since then, you encouraged me to exert myself to enroll in this PhD program, supporting me every step of the way up until now. I could not have completed this PhD without your continuous support. I am immensely appreciative of all of the support you have given me in paper writing, advising me with research ideas and ensuring that I do not lose track of enthusiasm when facing multiple paper rejections. Additionally, I am grateful for the Research Internship opportunity you provided me at A*STAR, from which I learnt a great deal. Your support has also helped me maintain a good standard of living as a 'married PhD student', which would have otherwise been very challenging.

I extend my gratitude to my dissertation committee members Prof. Jing Jiang , Prof. Kotaro Hara and Dr. Nairan Zhang. I am grateful for all the constructive feedback you provided through my Qualifying Exam, Dissertation Proposal, and final thesis. Your constant assessment has pushed me to do better and improve the quality of my dissertation.

Since my stint as an intern, I've accrued a debt of gratefulness to numerous friends at SMU. First and foremost, I thank Kasthuri, whom I always hold in high regard as an inspiration and someone who catalyzed the venture on this PhD journey. I extend my thanks to Meera, Sougata, Kenny, Amit and Vu, who I can always look up to as my seniors. Finally, I thank Prof. Thivya, Tuan, Lakmal, Vithurson, Gihan, Anuradha, Andrew, Darshana, Hemant, Dulaj for their unwavering support at various times and that my Ph.D. journey, miles away from home, felt like being at home. I wish to acknowledge Dinuka, Janaka, Dilan, Kasun, Charith, Shamil, Manusha, Saadia, Isuri, and Nuwan for their friendship, which fostered a homely atmosphere and uplifted spirits, alleviating the loneliness that often accompanies studying away from home.

I wish to express my appreciation to LiveLabs, LARC, MoE and M3S for their ongoing financial support, which has sustained me from my internship until now. Thank you to SMU administrative staff, Pei Huan, Caroline, Chui Ngoh, Chew Hong, Lydia, Nikki, Veronica, Desmond and Andrew. They were all very attentive and prompt with all my administrative requests.

Finally, I want to convey my heartfelt appreciation to my family for their unconditional love and support. Amma, Taththa, and Nangi, your support was invaluable when I had to spend so much time away from home to embark on this challenging journey. Special thanks are due to my wife, Ishanthi, who made silent sacrifices by pausing her career to come and support me throughout my PhD journey. Since you arrived in Singapore, my journey has been marked by numerous achievements, all of which have been facilitated by your presence and support.

To my beloved Ishy, for always being with me through my journey

&

To my parents and my sister, for your unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Multi-modal Instruction Comprehension for Human-AI Interaction Tasks

In recent years, there has been a surge in the popularity of AI agents capable of engaging in natural interactions with humans. This trend is notably fueled by the proliferation of consumer mobile and wearable devices equipped with a variety of sensors supporting capture of verbal, environmental and gestural cues. For instance, the latest Apple iPhones are equipped with LiDAR sensors designed specifically for augmented reality (AR)-based interactive applications. Similarly, advanced smart glasses like Microsoft Hololens [9] come equipped with a range of sensors, including eye-tracking, head tracking, depth sensors, inertial measurement units (IMUs), and cameras, all aimed at enhancing human-AI interactive applications. The integration of these interactive sensors into consumer devices paves the way for the evolution of traditional voice-based conversational agents, such as Apple's Siri and Amazon's Alexa, into multi-modal agents that leverage a combination of language, gesture, and visual information. A key capabality required for these supporting human-AI interactive applications is to understand human instructions comprising of a combination of language and gestural cues.

The foundation of AI-based human instruction comprehension lies in Deep Neural Network (DNN) models trained specifically for this task [129, 121]. Additionally, *multi-modal instruction comprehension* models [27] have also been enhanced to incorporate additional sensing modalities such as pointing gestures. However, these models are often resource-intensive and may not be suitable for deployment on pervasive devices, which typically have limited computational resources. Our key objective in this thesis is to support lightweight human instruction comprehension in pervasive devices with low latency and low energy overheads. Consequently, there are two major approaches aimed at adapting existing instruction comprehensions models to be lightweight enough for pervasive and consumer mobile devices.

1. **Static model optimization:** Static optimizations target to reduce the complexity of instruction comprehension DNNs by shrinking the size of the network. Traditional pruning approaches [118, 19] on DNNs can be typically applied for this scenario.

2. **Dynamic model optimization:** Dynamic optimizations target to judiciously reduce the complexity of the DNNs with respect to the input instruction. Broadly, this refers to the scenario where the execution pipeline changes for different input instructions.

In this thesis, we delve into both optimization techniques for reducing latency and power consumption in instruction comprehension, but the primary emphasis is on dynamic model optimizations. This choice is driven by the fundamental premise that ***natural human instructions, issued in a variety of field environments, are associated with comprehension tasks of varying complexity***. In addition, the variation in complexity can be associated with different modalities: e.g., one comprehension instance may involve a very complex verbal instruction, including deictic expressions [74], but a relatively uncluttered visual layout, whereas another instance may involve a much simpler verbal instruction but require the processing a highly cluttered visual environment with multiple confounding objects [87]. Consequently, we hypothesize that AI agents designed to comprehend these instructions do not need to adhere to a uniform execution pipeline for different task instructions. As a result, I primarily focus on developing diverse dynamic optimization techniques that have the capability to adapt their execution pipeline based

on the inherent complexity of the task instruction.

To support human instruction comprehension, the work in this dissertation draws on the foundational work on Visual Grounding or Referring Expression Comprehension (REC) models [129], primarily designed for interpreting verbal instructions within a given scene image. I extend these models to incorporate pointing gestures using depth images and subsequently propose various static and dynamic optimization techniques aimed at minimizing latency and energy consumption. In part I, the initial focus is on enhancing the efficiency of Image-based REC models through the integration of multi-modal inputs and the development of optimization strategies targeted to reduce latency and energy overheads.

Subsequently in part II, the research shifts towards Video-based Referring Expression Comprehension (VREC). Unlike static REC, VREC involves the comprehension of human instructions in relation to a video stream, i.e., a sequence of images.. These interactions require both spatial and temporal understanding, leading to the term Spatio-Temporal Video Grounding (STVG). Similar to the challenges encountered in the static REC problem, I explore several optimization techniques with the goal of jointly reducing latency and energy usage to facilitate the execution of VREC on pervasive devices. This extension addresses the dynamic and multifaceted nature of human instructions in video-based scenarios.

1.2 Motivating Scenarios

From section 1.1, it is evident that multi-modal instruction comprehension models will be beneficial for several human-AI interaction tasks. To further motivate this case, I illustrate two motivating scenarios in different contexts.

1.2.1 Motivating Scenario 1: Robot Assisted Aircraft Wheel Change

Consider an airport of the future that is looking to integrate latest technologies, such as wearable Augmented Reality (AR) devices and robots, to improve its operational productivity and safety. As a hypothetical but more concrete example, consider the goal of robot-assisted aircraft wheel changes. Traditionally, this task is labor-intensive, typically requiring 3-4 workers due to the substantial weight of an aircraft wheel, approximately 250KG. The aim is to transform the operation to a human-robot collaborative paradigm, where a single supervisor will provide instruction to one or more robots, which will perform the physically demanding tasks such as wheel mounting and dismounting. As depicted in figure 1.1, a worker wearing a smart glass instructs the robot, for instance, saying, '*Please align the wheel and screw the wheel cover.*' The smart glass captures the instruction, the worker's ego-centric viewpoint, and accompanying gestures, sending this information to the robot for comprehension. Additionally, the smart glass may display information from critical sensors attached to the aircraft and wheels during the wheel change operation. Given the stringent scheduling of maintenance tasks, there is a high priority on reducing the overall time required to complete the wheel change operation. Therefore, a system proposed to handle this scenario will need the following key capabilities:

1. **Multi-modal instruction comprehension:** The AI models embedded in the robot must robustly comprehend complex verbal instructions, gestures, and visual information provided by the worker through the smart glass. This capability is essential for accurately executing the intricate wheel change operation.

2. **Latency-aware execution pipeline:** The primary objective of the latency-aware execution pipeline is to preserve the interactive nature of co-working, but not to reduce the overall maintenance time. It is worth noting that the maintenance process typically spans around 15-20 minutes, rendering latencies of 10-15 seconds potentially inconsequential. However, it's imperative to maintain a low comprehension latency, ideally within the range of 500-1000 milliseconds. This ensures the responsiveness of human-robot interactions, preventing scenarios where the robot remains idle for prolonged periods, such as

Figure 1.1: Robot Assisted Aircraft Wheel Change

10 seconds, while processing instructions.

1.2.2 Motivating Scenario 2: Shopping Assistant

Sarah, the owner of a shopping mall, is planning to transform it into a smart shopping mall to enhance customer experience and improve business efficiency. Her focus is on deploying an assistive robot that aids customers with tasks like locating and retrieving specific products within a store in the mall. Additionally, she is intrigued by the possibility of introducing a smartglass-based AR application, that customers can use to interact with a set of assistive robots within her store. As shown in figure 1.2, a customer wearing the smart glass might issue a verbal instruction such as, '*Please pick up that brown-colored book in the middle of the second shelf.*' , while pointing at the desired book. The smart glass captures the scene image, depth image, and audio, which are then utilized by comprehension algorithms to identify and localize

the desired object within the mall. Once identified, the assistive robot retrieves the requested product and returns it to the customer. Additionally, the smart glass application may display product information on the screen.

To meet Sarah's expectations of a seamless system that reduces customer waiting times and optimizes workforce utilization even during busy periods, the proposed system should possess the following key capabilities:

1. **Multi-modal instruction comprehension:** AI models deployed in the assistive robot should comprehend a combination of visual, verbal and gestures provided by the customer captured through the smart glass.

2. **Efficient Human-Robot Interaction:** The system should enable efficient interaction between the smart glass user and the assistive robot, allowing for smooth communication of instructions and feedback. This ensures a seamless shopping experience for customers. This will require our proposed dynamic optimization approaches which will adjust its pipeline based on the user's query to provide a lower latency (on average), more responsive interaction experience.

3. **Adaptability to Busy Environments:** The system should be capable of handling increased customer traffic during busy times, where the robot may have to handle multiple customer requests at once. In such situations, the execution pipeline deployed in a single computing device within the robot should dynamically adjust based on the complexity of the user's requests. This dynamic adaptation ensures that the system can efficiently manage tasks, optimizing the utilization of resources to handle multiple customer requests concurrently.

Figure 1.2: Shopping assistant scenario

1.3 Design Goals

My primary goal in this dissertation is to develop and validate AI-based instruction comprehension algorithms for human-AI interaction tasks. For this, I would extract motivation from natural human-human interaction and consider the following design goals.

1. **Multi-modal sense-making:** Natural human-human interactions usually occur through a multitude of modalities such as visual, verbal and gestural cues. Thus, my focus is to develop comprehension algorithms to be supportive of multiple sensing modalities.

2. **Low latency sense-making:** I consider the scenario, where these AI agents require to be deployed on pervasive devices. Generally, pervasive devices are limited by computing resources. Thus, AI models need to be optimized for supporting low-latency processing with reduced resource usage, while achieving a comparable accuracy in sense-making. Low latency processing is pivotal in human-AI collaborative tasks to enable seamless interaction between human and the AI agent. To ensure that the comprehension models

are able to respond to human instructions promptly, without appearing to be idle for prolonged periods, we expect to achieve comprehension latencies $\leq$ 1 second with only $\leq$ 5% reductions in accuracy compared to a baseline un-optimized model in a representative pervasive device.

3. **Battery powered sense-making:** Pervasive devices are typically battery-powered, making it crucial for energy-efficient sense-making. Thus, I explore the possibility of reducing a) *sensing energy* and b) *processing energy* through various optimization techniques so as to significantly extend the operating lifetime of the devices, while still supporting the execution of complex DNN models for such multi-modal instruction comprehension. Generally, the faster a model operates, the lower its processing energy consumption. While we consider a latency of 1 second sufficient to ensure that the model is adequately interactive, further reduction in processing latency is desirable as this will further increase the operational lifetime of the battery-operated pervasive device. Thus, the goal is to maximize the reduction in the model latency while maintaining a minimal $\leq$ 5% reduction in accuracy. Additionally, I plan to selectively activate power-intensive sensors, such as LiDARs, only when necessary. Our goal is to minimize sensing energy consumption while maintaining the task accuracy within a $\leq$ 5% reduction.

In this thesis, the aforementioned design goals lack empirical support from user studies. For instance, we assume that a latency of $\leq$ 1sec is sufficient for enabling the desired interactivity of instruction comprehension models. Additionally, we assume that a reduction in accuracy within the range of $\leq$ 5% is acceptable in the specified motivating scenarios. It is important to acknowledge that these constraints are specific to each application and ideally should be validated through separate user studies conducted for each use-case. However, in my thesis, the focus is on devising a set of core approaches to reduce the computational latency and energy overheads of such machine comprehension, which I believe represent foundational advances that will benefit a variety of applications.

1.4 Thesis Statement

Comprehending human instructions is vital for effective Human-Robot collaboration. Human issued instructions that refer to a target object in a given scene typically convey information about the target using a combination of modalities such as language, gestures etc. The complexity of processing such instructions can vary widely depending on factors such as the specificity and verbosity of the references uttered in the language instruction, the scope for error in resolving the pointed location and the visual clutter in the scene. Mainstream Deep Neural Networks (DNNs), that perform this complex vision-language task using RGB camera data and text inputs, are computationally demanding and do not sufficiently exploit alternate sensors for integrating pointing gestures and the visual context. In this thesis, I extend these DNNs to incorporate gestural input using depth sensors and dynamic environment cues using event-based sensors in tandem with verbal input, and further demonstrate that it is feasible to significantly reduce their run-time resource footprint, thereby reducing latency and energy, by adopting complexity-aware techniques that exploit the redundancy or correlation in cross-modal information when processing instructions referring to objects in images as well as videos.

Initially, Chapter 2 provides an in-depth review of the literature related to the broad topic of multi-modal human instruction comprehension, spanning topics such as visual grounding, gestural comprehension and Video Grounding.

This thesis is then structured into two parts. *Part I: Image-based Human Instruction Comprehension* delves into the study of model optimizations applied to REC models, which analyze a single static image alongside language and, optionally, gestural modalities. In *Part II: Video-based Human Instruction Comprehension*, we broaden our methodologies to tackle more complex scenarios involving videos as vision input, moving beyond single static images.

In Chapter 3, we initially underscore the importance of pointing gestures in improving task accuracy for object acquisition tasks. To demonstrate this, we introduce and discuss the M2Gestic system [106], which integrates neural-based text parsing with a novel knowledge-graph traversal mechanism across multi-modal inputs encompassing vision, natural language text, and

pointing. Through a series of user studies, we establish that M2Gestic: a) achieves performance close to human levels when instructions are unambiguous, and b) demonstrates a significant (30%) enhancement in accuracy by incorporating hints from pointing gestures when instructions are ambiguous. This highlights the critical role of pointing gestures in disambiguating task instructions and improving overall accuracy.

In Chapter 4, we apply the paradigm of dynamic optimizations into vision and language based REC models. To this end, we develop SoftSkip [108], a multi-modal pruning strategy tailored for REC tasks. SoftSkip uses language features to dynamically skip computations in a REC model. Key to this approach is a paradigm termed *soft-skipping* where a computational block that is decided to be skipped by the pruning model is *approximated* with a light-weight computation. We reveal why such approximation, in contrast to the extant dynamic optimization techniques that completely skip some portions of the DNN execution pipeline, is crucial for such multi-modal REC instructions. To demonstrate, we apply this approach to RealGIN; a single-stage REC model and introduce LGMDP model to show 33% savings in latency while suffering a 0.5% loss in comprehension accuracy.

In chapter 5, we extend the dynamic optimization paradigm to REC tasks involving the fusion of vision, language, and pointing gestures. Our primary objective is to address the challenge of executing multi-modal REC models on pervasive devices while minimizing latency and energy consumption. To tackle this issue, we introduce the COSM2IC approach [107], an optimization technique for executing REC models on pervasive devices. In this work, we first enhance a baseline REC model to accommodate pointing gesture modality using depth sensor input. Additionally, to assess the performance of multi-modal REC, we curate a COSM2IC dataset, where a human instructor equipped with a Hololens [9] device issues table-top object acquisition instructions along with pointing gestures. Our optimization approach, COSM2IC, utilizes a Task Complexity Predictor (TCOP) to selectively invoke different pipelines based on input complexity. Leveraging this model-switching platform, we achieve a three-fold reduction in latency while maintaining comparable comprehension accuracy.

In Chapter 6, our focus shifts from solely reducing processing energy overheads, as seen in

SoftSkip and COSM2IC, to adopting a more comprehensive approach targeting both sensing and processing energy overheads. As demonstrated in COSM2IC, the integration of additional sensing modalities, such as LiDARs, enhances task accuracy but also introduces a significant increase in sensing energy overhead, thereby contributing to overall energy consumption. To address this challenge, we introduce a joint dynamic optimization technique called Commit-And-Switch (CAS), designed to concurrently reduce both sensing and processing energy overheads while maintaining comparable task accuracy. In CAS, we initially commit to a computational pipeline involving low overheads and a subset of available sensors. The task context estimated by this pipeline is then utilized to optionally switch to another energy-intensive DNN pipeline and activate additional sensors. By implementing our CAS paradigm, we introduce an optimized REC model termed RealGIN-MH for multi-modal target acquisition tasks, which achieves a 12.6x reduction in energy overheads while surpassing baseline dynamic model optimization approaches.

Until now, our proposed methodologies have been tailored for REC models operating solely on single static images. However, it's evident that not all task instructions can be sufficiently represented by a single static image. Some instructions may involve a moving object, an action performed by another user, or the user pointing to multiple objects within a single instruction. In Part II, we address these dynamic scenarios by extending our approach to Video Referring Expression Comprehension (VREC) models, optimizing them through our dynamic optimization paradigm.

In Chapter 7, we present GRefExSel, a VREC model designed to dynamically select a set of *anchor-frames* from the raw video for executing a complex VREC pipeline. This approach targets the reduction of latency and processing energy for pervasive deployments. In chapter 8, we further refine GRefExSel by introducing NeuroViG. NeuroViG jointly optimizes for both sensing and processing overheads. A notable feature of NeuroViG is the integration of an event camera with significantly lower sensing overhead into the VREC pipeline. This event camera acts as a trigger sensor, selectively activating the RGB camera only for the chosen anchor-frames, thereby reducing the overall processing overhead by approximately four times.

Finally, in Chapter 9, I conclude the thesis with a summary of my key findings as well as an enumeration of open challenges that provide directions for future work.

Chapter 2

Literature Review

2.1 Comprehending Natural Language Instructions for Human-AI Collaboration Tasks

The task of 'grounding' natural language instructions in AI agents involves parsing the instruction to extract phrases and assigning a direct meaning to them in the context of the real-world perceived by the AI agent. [81] have introduced probabilistic models to achieve grounding in a table-top manipulation setup involving objects that can handle spatial references and abstract concepts of cardinality (group of 2 blocks on the left) and ordinality (2^{nd} block from the left). Interactive dialog-based approaches have been considered for disambiguation of natural language instruction by the INGRESS system in [90] and the interactive text2pickup network in [14]. Reasoning-based systems, that recognize individual objects in the scene and perform high-level reasoning to answer verbal questions (e.g., answering "how many blue blocks are behind the red block?") have been proposed in [49, 120]. However, the instructions studied in these works are primarily uni-modal and they do not consider the perspective and scene ambiguity challenges. To tackle these limitations, we propose several multi-modal approaches that also incorporate pointing gestures as an additional modality. To investigate the impact of scene ambiguity and perspective ambiguity, in Chapter 5, we introduce a more rigorous dataset

named COSM2IC for table-top object acquisition instructions.

2.2 Ambiguity in Natural Language Instructions

In complex scene environments, natural language instructions are often associated with two types of ambiguities; a) block ambiguity, where there are multiple objects matching the same language instruction and b) perspective ambiguity, where the instruction could refer to both instructor or instructee perspective. Using a carefully selected set of table-top block arrangements that induce varying levels of ambiguity, [87] collected a large set of human-generated instructions to pick a particular block. They then quantified the effect of instructional ambiguity by asking other human subjects to interpret these instructions. Using this publicly-released, 'collaborative manipulation corpus', [59] showed that instructions that do not involve perspective references suffer from poor human comprehension. In Chapter 3, we presented a neuro-symbolic approach for comprehending these ambiguous instructions. This approach utilized a natural language parser to break down verbal instructions into a series of sub-programs. These sub-programs are then used to comprehend and identify the most probable target object that matches the instruction.

2.3 Referring Expression Comprehension (Visual Grounding)

Referring Expression Comprehension (REC), also known as Visual Grounding (VG), pertains to the task of identifying the bounding box coordinates of a target object referenced by a language instruction within an image. As this task involves spatially localizing an object within an image frame, it is often referred to as Spatial Grounding. In this dissertation, we propose that REC models can effectively address object acquisition tasks due to their strong synergy with this domain.

Although REC problem has been studied for a while, deep learning-based approaches gained ground after the release of the large-scale ReferIt dataset [52] generated using a crowd-sourced two-player game. To generate ReferIt samples, one of the players views an image and writes an expression that refers to the target object, while the other player uses this expression to click on the relevant region in the image. This approach was later extended to include images in the MSCOCO [63] dataset, with the instructions then as RefCOCO and RefCOCO+ datasets [122]. Using the ReferIt dataset, the authors also studied the visuo-linguistic characteristics of referring expressions and showed how visual attributes in the image input correlated with words used in the verbal instruction (e.g.,'Big' is most commonly associated with larger target regions whereas 'small', tiny', little' etc are associated with smaller regions).

Initially, deep neural models for REC were based on the CNN-LSTM [72, 122] framework, where the LSTM takes a word vector at each time step and attempts to match it with CNN-based visual features extracted from a candidate region within the image. These models adopt a max-margin-based training method for the LSTM such that the probability of referring expression is higher for the referred image region. Many of these early works showed that accurate modelling of contextual information was critical for achieving effective target inference. For example, Yu et.al. [122] used visual differences between objects to represent the visual context, achieving higher comprehension accuracy than [72]. Subsequently, [45] used whole-image CNN features to represent the context, while [76] proposed the use of use multiple-instance learning for effective context modelling. With the advent of attention mechanisms, MAttNet [121] introduced a modular framework that further enhanced REC accuracy. In Mattnet, separate modules, each with their individual attention mechanisms, utilize features of object locations, context or relationships.

Such multi-stage pipelines, however, are computationally prohibitive due to the need for a separate region proposal network. To address this, recent *single-stage* neural approaches [129, 116, 86, 117] have replaced the region ranking with a multi-modal bounding box regression stage. Yang et al [117] proposed an approach based on YOLOv3 [84], where they obtained a visual feature pyramid via the Darknet-53 [84] backbone network, and the language features for tex-

tual referring expression via BERT [33]. The visual feature pyramid is then concatenated with the verbal features at each level, and subsequently combined with a normalized spatial feature, to execute the bounding box regression. The authors recently extended their work further [116] by including a sub-query learning and modulation framework that decomposes long textual descriptions into shorter sub-queries; they demonstrated that recursive use of such visual features improved the ability to resolve ambiguity associated with longer verbal instructions. The Zero-Shot Grounding (ZSG) method [86] extracted image features by combining a ResNet (instead of Darknet) backbone with a feature pyramid network (FPN), while using a Bi-LSTM to extract the language representation. Zhou et al. [129] further extended ZSG to develop the RealGIN model. RealGIN includes a separate Adaptive Feature Selection (AFS) method that uses textual information to identify REC-relevant visual features and a multi-modal global attention mechanism named GARAN, and achieved 30FPS processing throughput (a 10-fold increase over MattNet) for REC tasks. Deng et al. [32] introduce a transformer-based method for Visual Grounding. More recently, in light of the success of Vision Language Models (VLM), researchers have repurposed VLMs for REC tasks, as demonstrated by CogVLM [102]. However, while these transformer-based models and VLMs excel in accurately comprehending referring expressions, their heightened complexity, significant latency, and demanding memory and processing energy requirements pose challenges for deployment on pervasive devices.

Building upon these efforts, we aim to achieve additional reductions in processing latency and energy consumption by implementing dynamic model optimizations for REC. Our fundamental approach involves employing an execution pipeline that dynamically adjusts its pathway for each input based on the task context. An exemplary illustration of this concept is the LGMDP model introduced in Chapter 4. This model leverages language features to predict the task context, thereby activating only the relevant modules in the REC model for the predicted task context. Through this strategy, we continue to diminish processing latency and energy in REC models.

2.4 Multi-modal Instruction Comprehension (Natural Language + Pointing)

A system that combines gestures and natural language for interpreting object references in a table-top setup was proposed by [74]. But in this study, the gestures were performed from just inches away from the target object. More recently, [109] proposed a real-time system that can identify one of four objects in a table-top setting involving common kitchen items, by combining the language references such as *hand me the bowl* as well as a pointing gesture which were performed from a few feet away. [53] proposed a system incremental resolution of multi-modal instructions (verbal + pointing). However, since the speaker generates the instructions from the same point-of-view as the listener, they do not consider the ambiguity due to perspective, which is highlighted as an important factor by [87]. Thus, the language as well as the table-top setup in [87], is much more ambiguous or cluttered than the other previous works, and hence we choose to build upon the work by [87] to address the challenges of accommodating ambiguity in natural multi-modal instructions in chapter 3. In chapters 5 and 6, we recognized the opportunity to reduce the size of REC models (Either through static or dynamic model optimizations) by leveraging additional pointing gesture information.

2.5 DNN Model Pruning

A broader body of research has tackled the problem of pruning DNN models, either statically or during run-time [118, 19, 20, 61, 101], to support efficient, low-latency DNN execution on pervasive devices. Efficient networks such as MobileNet and ShuffleNet embody the principle of static model compression, where redundancy is identified and eliminated in the channels, weights and filters in a complex visual backbone such as the VGG16, ResNet etc. Approaches such as [61], SkipNet [104] and Dynamic Convolutions [101] are examples of run-time pruning, where unwanted computations are eliminated during inference time with minimal impact on overall accuracy. Reinforcement learning is typically used to train the modules that decide

on such run-time, input-dependent skipping of specific computational blocks. All extant approaches, however, tackle only uni-modal neural models; refactoring them to our multi-modal REC task is non-trivial due to the need to preserve relevant contextual and relational information across the different modes.

To overcome the constraints of existing DNN model pruning methods applied to human instruction comprehension, we primarily proposed dynamic model optimizations in this thesis. These optimizations strategically adjust the execution pathway by predicting the task context or complexity through a shallow, low-processing overhead computation.

2.6 Dynamic Sensor Triggering

Well before the advent of DNNs, the concept of dynamic sensor triggering as a means for energy-efficient pervasive sensing was used by a variety of works, such as EEMS [105], Mercury [67], SensLoc [54] and E-Gesture [80]. These systems activate energy-intensive sensors on demand, only when features computed from data segments of a low-energy sensor stream indicate a specific context (e.g., likely high inference error in EEMS or the start of a gesture in E-Gesture). In contrast, Jigsaw [68] and SeeMon [51] focused on reducing the inference overhead by invoking the execution of the context classifier only when the computed features diverged significantly from previous values. Gordon et al. [38] used a prediction of potential future output states to dynamically select the sensor subset accordingly. The ACE middleware [78] introduced the concept of inference caching, where correlations between different activity contexts were used to infer a new context attribute indirectly without actual sensing. More recently, the Annapurna automated food diary system [88] used cheaper inertial sensors to dynamically trigger the expensive camera sensor on a smartwatch at selected stages of an ongoing eating gesture. All these systems are characterized by either the use of a single low-energy triggering sensor (often the accelerometer) or a reasonable latency tolerance for dynamic sensor activation, mostly because the activity context being captured is somewhat long-lived (e.g., driving vs. at-home in ACE) or repetitive (e.g., eating in Annapurna). In contrast, for human

instruction understanding, the triggering context itself (scene/instruction complexity) requires the use of DNNs ingesting multiple sensor data streams. For instance, in the CAS framework introduced in chapter 6, the triggering mechanism must discern the appropriate triggering event based on the encoded visual features captured through a DNN pipeline. Moreover, the framework must also be adept at capturing and interpreting short-lived pointing gestures without omission.

2.7 Temporal Grounding

Early approaches to Spatio-Temporal Video Grounding typically treated the problem as two distinct steps, namely spatial grounding through REC models and temporal grounding. Temporal grounding is the task of identifying the frames in a video that are relevant to a given linguistic expression. Initial approaches to this problem adopted a variety of strategies to obtain candidate temporal-segments called 'moments' within an untrimmed video. The Moment Context network (MCN) [16] used a heuristic approach while TALL [37] used a sliding window based approach [37] to obtain candidate moment proposals. Realising the importance of the quality of these proposals for grounding performance, later approaches [112, 111] developed dedicated networks for generating the moment proposals. The visual features from the proposed moments and the features from the linguistic query are then used to identify the best matching 'moment' for a given query. Separately, more efficient single-stage networks were pursued to perform candidate moment generation and matching score generation in one pass. To achieve this, several strategies such as anchoring [24], sampler networks [123] and segment-trees [127] have been attempted. Proposal free approaches such as L-Net [25] and DRFT [26] have also been developed such that they can directly predict the start-frame and end-frame of the relevant moment using end-to-end neural networks. These approaches first obtain a query-aware representation of the given video and then directly learn to predict the start and end frames. Some proposal free approaches have also used a reinforcement learning paradigm [42, 103].

2.8 Spatio-Temporal Video Grounding (STVG)

The research work on visual grounding then moved on to models that can simultaneously perform *spatio-temporal* video grounding (STVG). In these works, a single model is used to obtain a spatio-temporal tube (a series of bounding boxes on subsequent frames) that matches the given linguistic expression. Zhang et al., [126] proposed a large-scale dataset named VidSTG along with a graph-based baseline model (STGRN) for the STVG task using a two-stage approach. This approach relied on region or tubelet proposals obtained in the first stage that were used by the subsequent stage to find the best possible match with the given linguistic expression. Their work showed that such STVG models have significant performance advantages over the strategy of using separate pipelines for the sub-tasks of temporal and spatial grounding (For eg. use TALL [37] or L-Net [25] for temporal grounding on the untrimmed video and then apply spatial grounding techniques such as GroundR [85], WSSTG [28], STPR [113] on the extracted segments).

Following the success of transformer based models in vision-language tasks [58, 95, 69, 94], recent works have turned to transformer-based models for the STVG task. The STVGBert [93] performs this task by extending the ViLBERT [69] into a new component called ST-ViLBERT that preserves spatial information that is usually lost in ViLBERT during pooling operation. This new component uses the cross-modal features from the video clip and textual query as the input to produce the bounding box for each frame as well as predict the starting and ending frames. STGVT [97] focused on a human-centric variation of the STVG task using a specialised dataset named HC-STVG where the target object is always a human. TubeDETR performs the STVG task by extending the MDETR [50] framework using temporal localization losses, slow-fast encoding, and space-time decoding. STCAT is another recent model that uses a DETR-based framework to achieve state-of-the-art performance on both VidSTG and HC-STVG datasets. In contrast to TubeDETR, STCAT considers the global video context and maintains a template mechanism in the query-guided decoder for explicitly modelling consistency. Notably, TubeDETR and STCAT do not use pre-generated object/tubelet proposals in

their pipeline.

Although existing works such as TubeDETR and STCAT have demonstrated state-of-the-art performance in comprehending complex video-based instructions, these models have yet to be evaluated for latency and energy consumption on pervasive devices. As we shall showcase in Chapters 7 and 8, these models are extremely computationally demanding to run on a pervasive device with the desired latency and energy consumption. To tackle this, we propose a dynamic model optimization, where we select a set of anchor/candidate frames to limit the complex transformer-based operations. Our key hypothesis is that the intended temporal and spatial reasoning can be confined to these candidate frames, while the outputs for the remaining frames can be interpolated from motion-specific features, reducing computational overhead.

2.9 Processing with Neuromorphic Event Cameras

With the introduction of neuromorphic event cameras, many vision-based applications that traditionally relied on RGB cameras are beginning to adopt event camera streams as inputs. Event cameras offer several advantages over frame-based counterparts, including exceptionally high temporal resolution (on the order of microseconds), a wide dynamic range, and low power consumption. Due to their asynchronous nature, numerous efforts have been made to process event-based input streams using spiking neural networks (SNNs) [98].

Looking at it from a different perspective, an event stream can be conceptualized as a 4D representation with coordinates (x, y) denoting the 2D location of the event, polarity p indicating the event's sign, and timestamp t depicting when the event occurred. Prior to integrating event streams into processing pipelines for various vision-based applications, such raw event data can be transformed into: a) Frame-based representations [131], b) Time-surface representations [91], or c) Voxel-based methods [82]. Recent advances in leveraging event cameras for sense-making have also extended to the fusion of RGB and event camera streams. For instance, Zhou et al. [130] proposed a frame-based multi-modal deep neural network (DNN) pipeline

that simultaneously processes both a conventional RGB camera stream and an event stream for tasks like moving object detection. The proposed *NeuroViG* system, detailed in Chapter 8, employs a frame-based representation pipeline to opportunistically trigger the RGB camera for the STVG task. This approach reduces sensing energy consumption by avoiding continuous RGB camera operation. Instead, it leverages features from the event camera for visual context whenever possible, thereby minimizing the reliance on the more power-intensive RGB camera.

Part I

Image-based Human Instruction Comprehension

Chapter 3

Gesture Enhanced Comprehension of Ambiguous Human-to-Robot Instructions

In this chapter, we first demonstrate the possibility of utilizing pointing gestures to improve the comprehension accuracy of natural human instructions to robotic agents in human-robot collaborative tasks. While dynamic model optimizations have not been implemented in this chapter, our primary objective remains to demonstrate AI-based human instruction comprehension. We aim to achieve this by integrating both language and pointing gesture cues to optimize accuracy. Pointing gestures are often employed by humans when referring to one or more target objects and can help narrow down the spatial area within which an AI comprehension model needs to 'localize' the human-issued instructions. On the flip side, human pointing gesture are not precise but are typically associated with a cone of uncertainty (in 3D space) about the individual's intent. We present M2Gestic [106], a system that combines neural-based text parsing with a novel knowledge-graph traversal mechanism, over a multi-modal input of vision, natural language text and pointing. Via multiple studies related to a benchmark table top manipulation task, we show that (a) M2Gestic can achieve close-to-human performance in reasoning over unambiguous verbal instructions, and (b) incorporating pointing input (even with its inherent location uncertainty) in M2Gestic results in a significant ($\sim 30\%$) accuracy improvement when verbal instructions are ambiguous.

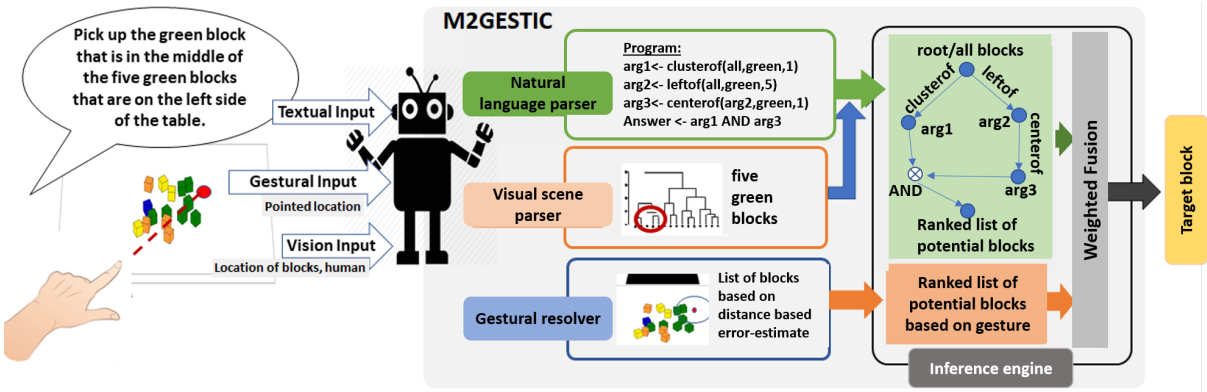


Figure 3.1: M2Gestic: System Components and Functionality

3.1 M2Gestic: Gesture Enhanced Comprehension

In section 1.2, we considered several scenarios of Human-AI Agent/ Robot interaction. To facilitate those scenarios and to maintain a natural interaction between the worker and robot, it is necessary to enable interaction via a mixture of multiple modalities, such as sight, speech *and gestures* (e.g., pointing). Therefore, supporting such natural human-robot interaction will require machine comprehension techniques that are *multi-modal*. Efforts on visual search and reasoning systems (e.g., [120, 49]) have explored the possibility of combining visual scene analysis with text understanding, albeit within fairly unambiguous task contexts. In contrast, we consider the possibility of multi-modal instruction comprehension for the *collaborative table top manipulation* task, where a robot attempts to interpret ambiguous "target acquisition" commands issued by a human. Using a benchmark dataset, Scalise et al. [87] assessed *human performance* in both generating and interpreting such visual perception-driven, natural language text instructions and demonstrated the challenge of *instructional ambiguity*. A sample table-top block-setup from this dataset can be seen in Fig 3.1 (on the left). Scalise et al. found that ambiguity resulting from the visual scene (e.g., many blocks with same attributes are closely packed) or imprecise perspective (e.g., does 'left' refer to your or my left?) affects accurate human comprehension of such language instructions.

In natural human communication, such verbal *instructional ambiguity* is often resolved via an accompanying gesture (e.g., *pointing*). Therefore, in this chapter we explore the design of machine comprehension techniques that tackle ambiguous table-top manipulation instructions,

by incorporating both *pointing gestures* and natural language text. Our premise is that a simple pointing gesture, overlaid on top of verbal and visual/scene analysis, can help reduce ambiguity significantly. While we confine our investigations to this table-top scenario, we strongly believe that, given past evidence on the importance of non-verbal cues, such as gestures, gaze and posture, in human-robot collaboration (e.g., [22] which looked at non-verbal responses *by* a social robot), our work has broader significance by *demonstrating the feasibility and benefits of incorporating gestural inputs in comprehending such ambiguous human→robot instructions*. We must, however, address two challenges:

(i) *Lack of Gestural Precision:* The pointing gesture itself is unlikely to be exact—humans may make an error between the pointed location and that of the intended target [35]. Additionally, it is likely that such errors may increase with distance from the target.

(ii) *Multi-Modal Fusion:* For automated machine comprehension, we will need a consistent mechanism to identify a target object given multiple sensory inputs. In particular, we must find a way to (a) parse and extract relevant spatial and/or descriptive attributes from the verbal command and use those attributes to *reason* over the object-level attributes (e.g., location, color, shape) provided by AI-based vision techniques, and (b) combine such reasoning with the potentially-erroneous, pointing-based spatial cues.

In this chapter, we address these challenges by (a) quantifying the nature of pointing-driven error in representative tasks and (b) developing a novel target selection mechanism that creates and parses a knowledge graph structure, based on multi-modal attributes generated by state-of-the-art deep learning techniques.

3.2 Preliminaries

We use the collaborative manipulation experimental setup [87] as our canonical use-case. This setup involves 28 different images of block arrangements (a typical block arrangement is illustrated in the top-left part of Figure 3.1), from which a *single* target block needs to be identified.

The arrangements and the corresponding target-blocks have been designed to generate different forms of ambiguity when human subjects generate verbal instructions to pick up the target block. In the data published by [87], each setup is also accompanied by a set of 50 different human-generated natural language text 'pickup' instructions to pick-up the target block. However, no gesture-related data is included.

Our primary goal is to develop an automated system termed M2Gestic (**M**ulti-**M**odal **Gest**ure-enhanced **I**nstruction **C**omprehension System), which combines verbal reasoning over visual content with accompanying pointing gestures for enhanced comprehension of multi-modal 'pickup' instructions. Note that M2Gestic does not aim to improve the technology for accurate tracking of the pointing gesture; nor does it focus on techniques for conversion of audio to textual input or performing object detection. Instead, it assumes the use of state-of-the-art systems to perform these perceptual tasks.

Figure 3.1 illustrates the components and overall workflow of M2Gestic. At a high-level, it consists of the following components (detailed descriptions are deferred to Section 3.3): (i) The *Visual Scene Parser* processes the table-top image (consisting of the blocks) to create a multi-attribute representation of the objects; (ii) the *Natural Language Parser* similarly processes the textual instruction, converting it into a set of machine-understandable primitives; (iii) the *Gesture Resolver* uses the pointing gesture to identify a subset of candidate blocks (based on the distance from the table) on the table-top surface, while the (iv) *Multi-Modal Inference Engine* fuses the inputs from these 3 previous components to perform target selection.

3.2.1 Empirical User Studies

To design and evaluate such a system, we shall utilize the following experimental studies (using a setup similar to [87]).

Study 1: Characterizing Pointing Gestures: We conduct this study (detailed in Section 3.2.2) to gauge the error characteristics of pointing gestures performed by human subjects, especially

Figure 3.2: User performing a pointing gesture towards a target block (in the projected image) using HTC VIVE

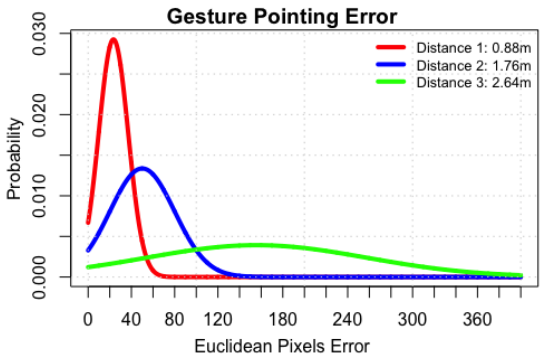


Figure 3.3: Pointing error distribution vs. screen-human distance

as a function of the distance between the human instructor and the target.

Study 2: Baselining Human Performance with Pointing Enhanced Instructions: - In this study (detailed in Section 3.4.2), a virtual 3D environment was developed to recreate the same ambiguous table-top setups used in [87] and a human avatar is shown performing a pointing gesture (with zero pointing error) towards the desired block from a distance. We thus created a new dataset of images that show the robot's view of the setup where both the pointing gesture as well as the blocks arrangement are visible. These images were provided to human subjects (recruited via the Amazon Mechanical Turk crowdsourcing platform) to infer the correct target-block by combining inputs from the text instructions with additional gestural (pointing) input. This serves as a upperbound of human comprehension capability (under idealized zero pointing error) against which to compare M2Gestic's performance.

We additionally employ the following experimental evaluations to compare M2Gestic's performance against these human baselines: (1) *Automated Comprehension without Gestures*: we evaluated the ability of the M2Gestic system to combine its Visual Scene Parser and Natural Language Parser to choose the right target-block, for each of the 28 images and corresponding 50 text instructions in the original dataset [87]; (2) *Automated Gesture-Enhanced Multi-Modal Comprehension:* we evaluate the improvement in block selection accuracy achieved by M2Gestic in the presence of such noisy *synthetically-generated pointing input*.

3.2.2 Study 1: Characterizing Pointing Gestures

Given our high-level goal of incorporating pointing input for multi-modal instruction comprehension, we first study and characterize the nature of human pointing input. In this study (illustrated in Figure 3.2), the images of each of the 28 block setups were projected on a screen and 15 human subjects were asked to perform a pointing gesture towards the specified target block using a calibrated HTC VIVE system [44]. A calibrated HTC VIVE is known to be a highly accurate (error $\leq$ 0.02cm) [79] in tracing the pointed location. This system uses two pre-calibrated cameras that help provide information regarding the pointed location on the screen. For the experiments, we set the two cameras 5 meters apart. The pointing gestures were performed by the subjects from three specifically marked positions (denoted p_1, p_2, p_3) in the room along a straight-line drawn from the center of the screen, that were $d = \{88cm, 176cm, 264cm\}$, respectively, away from the screen. We chose these 3 distances because of the technical limitations of the VIVE System: the two VIVE cameras need to be separated by a diagonal distance of less than 5m to ensure that the VIVE controller is track-able by the cameras. Given this limitation, we chose 3 equidistant points between the screen and the maximal distance (264cm) that allows the controller to remain detectable. To provide operational familiarity with, and perceptual calibration on, the VIVE system, each subject had a training period (of a few minutes) where the pointing cursor was 'on'–i.e., the subjects could receive real-time visual feedback about the pointed location on the screen–and were asked to specifically 'target' the 4 edges of the screen. To mimic the real-world environment (such as a smart factory floor) where the human instructor will not have any such visual feedback, the cursor was, however, disabled during the actual 'pointing' study. The image-setups were shown in randomized order and each image-setup was shown to each subject thrice. Thus a total of $15 \times 3 \times 28 = 1260$ pointing gesture data were collected for each position p_i.

Fig 3.3 shows the probability distribution of the gesture pointing error ($\delta_i, i = \{1,2,3\}$) across all users, for the 3 distances $\{p_1, p_2, p_3\}$. We observe that: (a) the average pointing error (in pixels) increases non-linearly with increasing d (average error= 23.43 pixels, 50.05 pixels, 155.76 pixels at $d = 88cm, 176cm, 264cm$ respectively), and (b) error variance increases with d

as well. Additionally, we found that the average *error angle*, subtended at the human's location, was $< 3^{\circ}$, across all 3 distances. We can thus conclude: *M2Gestic's Inference Engine must be able to tolerate moderate errors in the instructor's pointing input, with the likelihood of such error being higher at greater human-table distances.*

3.3 M2Gestic: System Overview

In this section, we describe the detailed design of the 4 key functional components of M2Gestic (illustrated in Figure 3.1).

3.3.1 Visual Scene Parser

The *visual scene parser* is responsible for generating a representation of the relative positions, and selected attributes, of the various objects in the image-setup. In our experimental setup, the objects in the scene are the blocks, the robot, the table and the human instructor. The blocks are all cube-shaped and have one of the four colours (green, blue, yellow or orange). The objects in this setup are fairly simple to detect using standard computer vision methods (e.g., using YoLo [83] or SSD [65]) and is not the subject of this chapter. We assume that we know the position of the center of all the objects. However, as mentioned in [87], the natural language instructions generated by human subjects often contain hybrid 'density-based' references such as "the blue cluster in the middle", "the three blocks near you" etc. which require a hierarchical understanding of the objects in the scene. Therefore, we use a hierarchical agglomerative clustering approach to enable understanding of such phrases. To achieve this hierarchical representation, the distance between each object pair, calculated from the co-ordinates of the centers, is used to perform agglomerative clustering. (In addition to such clustering, the visual parser annotates each object with its 'color' and other relevant attributes such as shape or texture.) As an illustrative example, consider the dendrogram shown in Fig 3.4. Now consider the phrase "5 green blocks that are on the left side of the table". From the dendrogram, the 5 marked

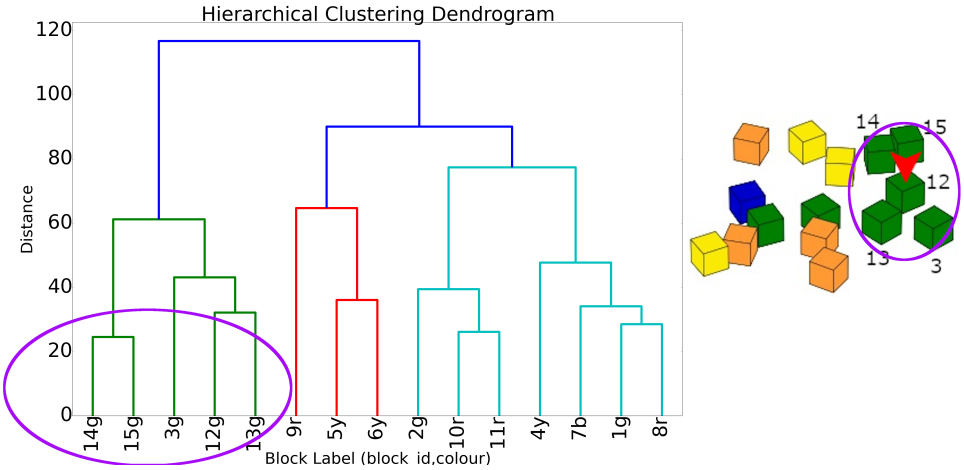


Figure 3.4: Hierarchical clustering dendogram to identify clusters of objects

blocks that are potentially referred to by this phrase can be immediately identified. Besides the hierarchical clustering, the parser also uses standard spatial reasoning techniques to create additional knowledge representations that capture: (a) the *perspective* information (e.g., closest/furthest/leftof/rightof from me/you), and (b) the *relational* information (e.g., pair-wise object distance, objects in the center, etc).

3.3.2 Natural language text parser

The natural language parser is responsible for converting the human-generated text instructions, describing the specific object to be picked up, into a computer program consisting of predefined functions. As the instructions typically contain one or more spatial prepositions denoting the relative positions of objects, the pre-defined functions typically correspond to spatial relations such as *leftof*, *rightof*, etc. A list of all the pre-defined functions used in M2Gestic and their respective descriptions are given in table 3.1

As demonstrated in [120], neural network techniques can be used to convert the entire natural language instruction into a sequence of such pre-defined functions. This is illustrated by the following simple natural language instruction, *"Please grab the yellow block that is the second from your far left"*. The corresponding structured program, which has just a single function call in this case, would be **farleft(you,yellow,2)**. In this example, *farleft(...)* is a function in

Table 3.1: A list of all the pre-defined functions and their descriptions

Pre-defined function	**Function description**
closest/furthest/nextto	find objects closest/furthest/next to
clusterof	find clusters of objects
leftof/rightof/topof/ bottomof/centerof/	find objects on left/right/top/ bottom/center
farleftof/farrightof	find objects from far left/far right

our robot's command vocabulary. The first input parameter 'you' refers to the perspective (i.e your left vs. my left). The second parameter 'yellow' says that only yellow blocks need to be targeted. The third parameter '2' specifies that we are looking for the 2^{nd} yellow block. Upon execution, this function will return a ranked vector of 15 elements (each element corresponding to one of the 15 blocks in the scene), with a lower rank implying a closer match. In our example, the rank will be the lowest for the 2^{nd} yellow block from the left of the robot. Other yellow blocks will get their ranks based on how close they are to the 'far-left' of the black-figure, with all non-yellow blocks assigned a rank=16 (the highest distance rank). We call this ranked list as a *sub-scene*, because this intermediate representation filters/prioritizes the blocks from the original/reference scene for subsequent operations. For recursive application, each of the input parameters may actually be specified as a previously-computed sub-scene. Thus, each function in our robot's vocabulary, is designed to take 3 input arguments, viz. (a) perspective/reference subscene, (b) target subscene and (c) rank/number. The output of the function is another subscene, that rank blocks based on this function's logic.

Now let us look at a more complex instruction - "*Grab the orange block that is furthest to the right and at the bottom beside a yellow block*", which results in multiple such structured robotic functions. The corresponding program is shown in (3.1).

$$\begin{aligned} farright(none, orange, 1) &= arg1 \\ bottomof(none, orange, 1) &= arg2 \\ and(arg1, arg2) &= arg3 \\ nextto(yellow, orange, 1) &= arg4 \\ and(arg3, arg4) &= ans \end{aligned} \tag{3.1}$$

The above example demonstrates the potentially recursive nature of such functions: output subscenes from a function may be used as the input subscene for another function (as exemplified by the two *and* functions).

Each such manipulation instruction can thus be converted into a sequence of functions, with additional {AND, OR, NOT} operators expressing the selection predicates. To establish a ground truth corpus, we first manually converted each of the 1400 instructions in the dataset into such programs. Subsequently, we trained a neural network model, as part of M2Gestic 'natural language parser' component, to generate such structured program syntax automatically from the natural-language text instructions. Inspired by state-of-the-art DNN-based machine translation techniques, we use the Attentional Recurrent Neural Network proposed in [70] to perform such a *sequence-to-sequence* mapping. Of course, such training requires a large training dataset. As the original collaborative manipulation corpus has just 1400 instructions, we *augmented* this dataset with additional instructions that are synthetically generated by changing the colours, perspective, words, phrases and instruction type. We also combined simple instructions in the original dataset to add more complex instructional examples to this training dataset.

Note that the original dataset contains several examples of ambiguous instructions that are typical of human-human conversations. For example, the instruction "*Please grab the yellow block that is the furthest*" suffers from perspective ambiguity: the target block could be the furthest yellow block **from the user** or **from the robot**. Similarly, the instruction "*Please pickup the topmost block*" shows ambiguity in both perspective as well as color attributes of the target block.

3.3.3 Gesture Resolver

We use state-of-the-art gesture/pose tracking systems to help track the arm movement/pose of the instructor's limb, and obtain an estimate of the table-top location of the pointing gesture. Based on this table-top location, we derive a gesture-based subscene, which is a ranked list for blocks based on the distance from the pointed position on the table-top. It is important to

distinguish between two distinct sources of pointing error: (a) the *intrinsic instructional error*, which arises from the fact that a human is unable to direct his pointing gesture *precisely* at the object that he intends to target, and (b) the *pointing tracking error*, which arises from the limitation/inaccuracy of the tracking technology. Note that M2Gestic's logic is independent of (b), and is primarily concerned with accommodating the error arising out of intrinsic human limitations. For our current implementation of M2Gestic, we utilize a calibrated (HTC VIVE) [44] tracker to provide an estimate of the human instructor's pointed location. While there is clearly tracking error, the pose estimation error is usually very small ($\leq$ 1cm) in such well-calibrated systems.

3.3.4 Inference Engine

The outputs from the visual scene parser, the natural language text parser and the gesture resolver are provided to the Inference Engine, which makes a decision on the target block. The program generated by the text parser may be represented as a tree structure whose individual nodes represent a *sub-scene* and the edges represent one of AND/OR/NOT operations. The various knowledge representations provided by the visual parser (e.g., object clusters, perspective relationships, object attributes and object-pair relationships) are used to execute individual functions of the program, and thereby generate a *ranked sub-scene vector of objects* for each node on this AND-OR-NOT tree. By traversing this tree from the leaves to the root, we can compute the final composite ranking vector, denoting the relative *fit* of individual blocks to the original instruction. We combine two ranked sub-scenes using a linearly-weighted formula, illustrated below for the AND operator. In the Eq. (3.3), consider R^1 to be the ranking vector of sub-scene 1, R^2 be the ranking vector of sub-scene 2 and let R^{ret} be the sub-scene obtained by combining R^1 and R^2 using the AND operator. Let $R^1 = \{R^1_1, R^1_2, \ldots R^1_k, \ldots R^1_m\}$ *and* $R^2 = \{R^2_1, R^2_2, \ldots R^2_k, \ldots R^2_m\}$ Then, the Rank-Sum s is given by

$$s = w_1 * R^1 + w_2 * R^2 \quad \forall k \in [1, m] \tag{3.2}$$

For purpose of generality, we define w_1 and w_2 as weights given to each sub-scene. For the current implementation of the text parser, we consider all sub-scenes to be of equal importance (i.e, w_1=w_2=0.5). However, in future, it might be possible to assign importance to certain parts of the sentence (e.g., using attentional mechanisms). This ranking vector is used as an input to the subsequent subscenes. If this ranking vector represents the return subscene, final output can be inferred from the indices of the blocks with the lowest rank-sum, given by $k_{opt} = argmin(s)$. (If the instruction is ambiguous there could be multiple k_{opt} values, otherwise there exists only one k_{opt} value). Let $k_{opt} = [k^1_{opt}, k^2_{opt}, ..., k^l_{opt}]$. Therefore the final return vector R^{ret} as a result of the AND operation can be obtained as below,

$$\begin{aligned} Let\ R^{ret} &= \{R^{ret}_1, R^{ret}_2, \ldots, R^{ret}_k, \ldots R^{ret}_m\}\ ; k \in [1, m] \\ if\ k \in k_{opt}\ then\ R^{ret}_k &= 1\ else\ R^{ret}_k = 0 \end{aligned} \tag{3.3}$$

A similar approach is used for the OR operation as well.

Extension to Incorporate Pointing Information: We apply the same 'weighted' approach (introduced in Eq. 3.3) to fuse the knowledge from pointing gestures. Let R^l be the final ranking vector obtained from the text and vision parsers. Given a pointing location, we can similarly obtain another ranking vector, where the ranks are sorted by the distance of each block from the pointed location. Let R^g represent this gesture-driven ranking vector. We can then apply the same reasoning outlined in Equation (3.3)–i.e., first compute a linear weight $w * R^l + (1 - w) * R^g$ for each object, and then select the object with the lowest 'distance rank'.

M2Gestic's gesture-fusion technique, takes into account the increase in the pointing uncertainty/error with an increase in the instructor-object distance. Because the pointing uncertainty is lower when the user is closer (and vice versa), we use a larger value of w (reduced importance to the pointing input) when the instructor-object distance is larger, and vice versa. In Section 3.4.2), we shall see that this 'weighted technique' proves vital to ensuring that M2Gestic's comprehension accuracy proves robust (and outperforms human performance) even with increasing distance.

3.4 Evaluation

We now present our evaluation results for instruction comprehension, comparing the automated M2Gestic system with the corresponding human perception performance, both with and without the added pointing input.

3.4.1 Text Instruction Understanding (No Gestures)

3.4.1.1 Accuracy of Text parser.

We first evaluated the accuracy of the Attentional RNN-based technique for converting verbal instructions to *programs*. We trained the natural language text parser model on the *augmented* dataset with $80\% - 20\%$ train/test split and obtained an accuracy of 99.7% (the accuracy was slightly lower (95%) on the original data). Note that the augmented dataset did not include the original 1400 instructions. This confirms the ability of M2Gestic's RNN to convert the natural language input into accurate machine-readable programs.

3.4.1.2 Block Identification Accuracy

Then we evaluated the accuracy of the overall M2Gestic system, where its Inference Engine utilizes *only* the visual and text parsing pipelines. The original dataset also classified 1400 instructions as ambiguous (626) vs. unambiguous (774), based on whether more than one block in the scene potentially satisfies the instruction's combination of block or perspective predicates. Table 3.2 (specifically, the two columns categorized under 'No Gestures') provides the results for this scenario, both overall and under the presence/absence of ambiguity. The results for human accuracy (73.62%, based on an Amazon Mechanical Turk study) are reproduced from [87]. We find that the automated M2Gestic approach achieves *human-comparable* performance (80.84%) for non-ambiguous instructions, but *exhibits dramatic performance degra-*

Table 3.2: Potential improvement in accuracy of system using weighted inference scheme

	No Gestures		**With Gestures**					
	Accuracy (Text only)		**Accuracy (d1=88cm)**		**Accuracy (d2=176cm)**		**Accuracy (d3=264cm)**	
	Human	**M2Gestic**	**Human**	**M2Gestic**	**Human**	**M2Gestic**	**Human**	**M2Gestic**
Ambiguous Inst.	64.79%	29.26%	70.18%	60.73%	63.71%	42.37%	60.99%	29.26%
Unambiguous Inst.	80.79%	80.84%	83.29%	83.48%	83.89%	79.06%	78.88%	80.84%
All Inst.	73.64%	61.12%	77.50%	74.75%	74.88%	65.14%	70.93%	61.12%
Only pointing	–	–	23.78%	21.43%	11.04%	3.57%	10.0%	0%

dation (accuracy= 29.26%) *in the presence of instruction ambiguity.* Clearly, machine comprehension requires additional cues (specifically, pointing input) to tackle such real-world instructional ambiguity.

3.4.2 Multi-modal Understanding (With Gestures)

We next quantify the added benefits provided by the inclusion of pointing input about the *likely* location of the target block.

3.4.2.1 Study 2: Human Performance With Pointing Input

We first quantified the ability of humans (thus, both providing a competitive baseline for M2Gestic) to use the combination of pointing gesture information and text instructions to infer the target block.

Experimental setup: For this study, we used a virtual 3D environment (using Unity 3D) to simulate the same 28 table-top block arrangements in the original dataset. However, the original images represented the view-point of the human instructor. Since the multi-modal inference is performed by another agent on the opposite side of the table, we transformed the images to represent the perspective of the agent performing the comprehension task. Figure 3.5 provides an example of this transformed perspective, which includes the table-top objects, as well as the pointing gesture made by the human instructor (the avatar in the figure). We generated such views (corresponding to the 3 different distances used in Study 1), by fixing the instructor's

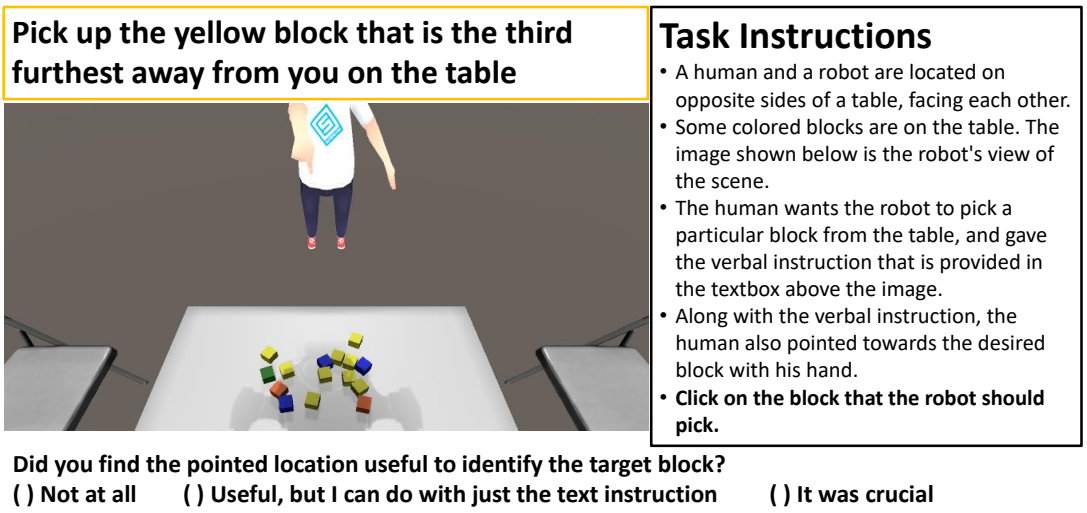


Figure 3.5: Study 2 - Setup used to study human performance in interpreting the instructions along with a gesture.

height at (190 cm), the block size to (5 cm) and the agent's height at (200 cm). To generate accurate pointing input, we adjusted the pose of the pointing hand of the human instructor (the avatar in Figure 3.5) to first point *exactly* towards the intended block by using a Unity-provided ray tracing model that can track and visually illustrate the pointed location. Then we took a screen grab of the resulting scene, *as viewed by the agent performing the inference*. 622 human-subjects, recruited via the Amazon Mechanical Turk platform, were then asked to use these pointing-included images, along with the text instructions, to infer the target block (as illustrated in Figure 3.5). Participants were also asked an additional question *"Did you find the pointed location useful to identify the target block"*, with one of 3 possible answers, viz. {'Not at all', 'Useful, but I can do with just the text instruction', 'It was crucial'} to help understand how human subjects assign more/less importance to the pointing gesture. Note that the pointing input for these human studies had *no error*; accordingly, the human perceptual performance provides the baseline under the most-optimistic gestural context.

Demographics: Each of the 622 Amazon Mechanical Turk workers were asked to perform at least 25 HITs. From the data collected we rejected the 'low-quality' assignments (39 workers) that matched any one of the following criteria: (a) Reject if accuracy $< 30\%$; (b) Reject if number of HITs done by participant < 25; and (c) Reject if user selected multiple points/objects. After rejections, each image was annotated by an average of 7 workers. The workers were

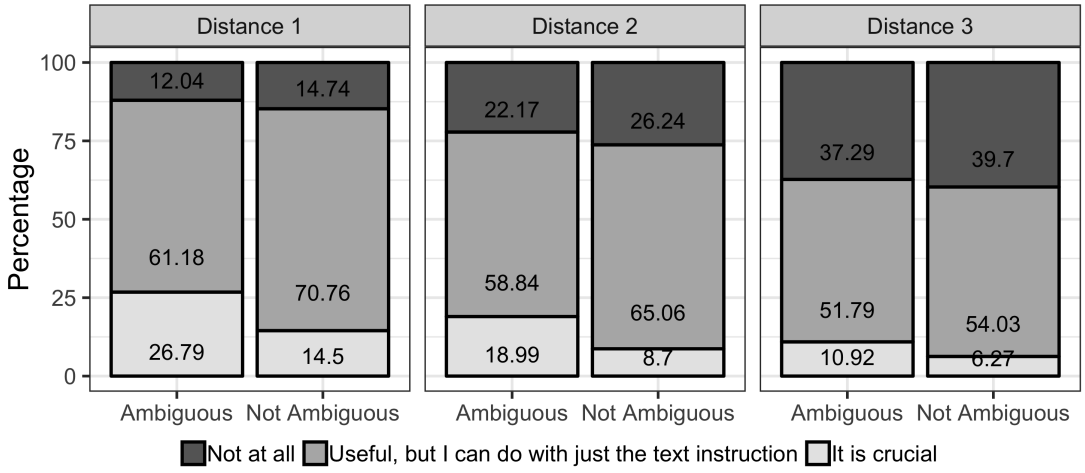


Figure 3.6: User perception of utility of pointing input

requested to provide three demographic details: 1) Age (73.51% in the $20-40$ age group), 2) Gender (46.5% male)and 3) Whether English is their first language (English=80.6%).

Human Performance: Table 3.2 (under the "With Gestures" heading) summarizes the average comprehension accuracy of both human subjects and M2Gestic. The last row of Table 3.2 provides the results when comprehension is performed *solely* using pointing input–i.e.,without parsing the text instruction. Even at a close distance of 88cm, the accuracy of human subjects in choosing the correct block based on pointing alone is very low (23.8%), when compared to using only the text instructions (73.64%); this accuracy drops by 10% when the instructor is 264cm away. Clearly, pointing gestures are insufficient for such cluttered table-top conditions, in contrast to earlier pointing based studies [75, 40], which use uncluttered setups.

More importantly, incorporating pointing input (in tandem with text parsing and visual scene analysis) improved human accuracy to 70.18% (5.3% higher than text parsing) for distance 1 (88cm away from the screen). However, human comprehension performance degrades with the instructor-object distance; in fact, at distance 3 (264 cm), the use of gestures actually causes selection accuracy to degrade below that achievable without gestural input! Clearly, while pointing can be beneficial, the 'blind' use of pointing input may be counter-productive if it is too noisy (performed from longer distances). The questionnaire responses, plotted in Figure 3.6, corroborate this insight: users perceived very low utility from the pointing information when the human instructor was farther away from the table ($d3 = 264$cm).

3.4.2.2 Performance of M2Gestic

We then evaluated the performance of M2Gestic, when its Inference Engine is provided the pointing data from study 1. *Note that M2Gestic does not, unlike the Amazon Mechanical Turk study, have an accurate pointing input, but assumes an error spread around the pointed table-top location.* For M2Gestic, the target selection accuracy improves dramatically (to 60.73% for the ambiguous instructions, as opposed to a baseline of just 29.26%) when pointing (from distance $d1 = 88$cm) is used to augment the textual instructions. For gestures from distance $d2$, the accuracy improvement for ambiguous instructions is still significant (about 13% higher vs. text-only). These results were obtained by empirically setting the weight factor ($w2$) values to $\{0.4, 0.2\}$, for distances $d1$ and $d2$ respectively. However, for distance $d3 = 264$cm, we observed that M2Gestic performed best with $w2 = 0$–i.e., when the pointing input was completely ignored, causing the performance to revert to its baseline ($\sim 29.26\%$ and 80.84% for ambiguous and unambiguous instructions respectively, in Table 3.2). Accordingly, similar to humans, the robotic agent should be capable of adjusting its fusion logic automatically, and discard pointing input (due to the likely large noise) if the instructor is too far from the objects. In addition, similar to the observation with human agents, the accuracy of M2Gestic is also very low (21.43% at $d1 = 88$ cm and 0% at $d3 = 264$ cm) when *solely* using the pointing input, further corroborating the limitation of pointing-only interactions in cluttered environments.

We also investigated the performance of M2Gestic with a weight of 0.5 for the gesture input. In this case we obtained lower accuracy values of $72.20\%, 62.43\%$ and 43.57 respectively, for distances $d1$, $d2$ and $d3$ over all the instructions in the dataset. Clearly, giving equal weightage to the pointing input is inadvisable and counter-productive. On further analysis, we found that pointing input helps resolve both perspective and block ambiguities from distance $d1$. At distance=$d2$, the pointing input still helped to solve certain perspective-related ambiguities, but not block-related ones.

3.5 Discussion

Considering the current advancements in VLMs/LLMs, we foresee several potential future directions for the proposed M2Gestic pipeline. One such direction involves leveraging an LLM to translate human instructions into a sequence of predefined functions, thereby bypassing the need for an Attentional Recurrent Neural Network. While Attentional RNNs were state-of-the-art models at the time of proposing M2Gestic, we anticipate that contemporary LLMs or transformer-based approaches offer significantly improved accuracy for similar tasks. However, it is essential to acknowledge that integrating such models may introduce higher latency and processing overheads, aspects that were not the primary focus of the proposed M2Gestic work.

3.5.1 Achieved Design Goals

With the proposed M2Gestic system, we achieved the design goal of *multi-modal sense-making*. Specifically, we incorporated pointing gestures in addition to verbal descriptions processed through the proposed neuro-symbolic approach in M2Gestic. However, M2Gestic has yet to achieve the other two design goals considered in this thesis. In the upcoming chapters, we will focus on achieving the other two design goals: reduced latency and energy consumption, while maintaining comparable task accuracy. This will be accomplished primarily by incorporating various dynamic model optimizations.

3.6 Summary

In summary, this work demonstrated the capability of AI-based human instruction understanding with a combination, visual, language and pointing gesture cues. Despite inevitable errors, the combination of pointing gestures and natural language text can lead to a significant improvement ($\sim$ 30% for robotic agents, and $\sim$ 5% for human subjects) in the accuracy of com-

prehending ambiguous human-to-robot instructions in our benchmark table top dataset [87]). This work made the following key contributions:

- *Develop a Multi-modal Target Selection Algorithm:* We describe a knowledge graph based technique for instruction comprehension, called M2Gestic (**M**ulti-**M**odal **Gest**ure-enhanced **I**nstruction **C**omprehension System, pronounced 'majestic'). M2Gestic combines (a) a neural (RNN-based) approach to automatically generate machine-understandable selection commands from natural language instructions, (b) a vision-based hierarchical clustering mechanism to represent salient spatial relationships under varying levels of clutter, and (c) a fusion mechanism that additionally ranks the 'fit' of objects based on their spatial alignment with the potentially-erroneous pointing location.

- *Quantify and Accommodate Pointing Gesture Error:* Through detailed empirical in-the-lab studies, we quantify the range of human error associated with natural pointing gestures. More specifically, we show that the distance error (at the table-top) increases non-linearly as a function of the human instructor's distance from the object (mean pointing error= 23.4 pixels at a distance of 88 cm, increases to 155.8 pixels at 264cm), which can imperil the usefulness of pointing input.

- *Establish the Efficacy of M2Gestic-based Comprehension, both with and without Pointing:* Using the benchmark table-top manipulation dataset [87], we first show that agent-based comprehension using text-only instructions (no pointing gestures) can achieve 61.12% accuracy in target-selection, compared to 73.64% accuracy previously reported for human respondents in [87]. Subsequently, using a series of in-the-lab and crowd-sourced studies, we demonstrate how the incorporation of *pointing input* (along with verbal and visual comprehension) helps improve this comprehension accuracy. From the realistic studies conducted using Amazon Mechanical Turk [15], with 622 respondents and 4200 unique task instances, we show that pointing input from a close distance enhances human comprehension accuracy from 73.64% to 77.5%, but exhibits a $\sim$5% drop when the instructor-object distance increases. For the M2Gestic-based AI/robotic agent, the comprehension accuracy of automated multi-modal comprehension (under

empirically-derived distributions of pointing error) on the entire dataset improves from 61.12% to 74.75% when the instructor is close to the objects. Moreover, this comprehension improvement is dramatic (30%) for the ambiguous verbal instructions. Finally, we show how a distance-weighted variant of M2Gestic provides *robustness*, ensuring that M2Gestic's performance, while suffering degradation, does not drop below the no-gesture baseline even when the instructor is at larger distances (implying larger spread of pointing error).

Chapter 4

SoftSkip: Empowering Multi-Modal Dynamic Pruning for Single-Stage Referring Comprehension

Starting from this chapter onward, our focus shifts towards optimizing instruction comprehension models for latency and energy overheads. For this, we leverage Referring Expression Comprehension (REC) models. These models, when provided with an image along with verbal and/or gestural inputs, identify the bounding box of the referred object. As motivated in chapter 1.1, our fundamental premise is that ***natural human instructions manifest varying task complexities***. Drawing inspiration from this premise, we introduce a dynamic optimization technique for Vision + Language REC models known as SoftSkip [108]. This approach aims to jointly reduce both latency and processing energy with only a minimal loss of approximately 1% in accuracy. The core principle of this method is to utilize language features as a pivot for dynamically and judiciously selecting, and/or reducing the complexity of the computational blocks that need to be executed.

4.1 Introduction

Initial approaches for REC adopted a three-stage process, viz., (Step 1) generate region proposals on the image, (Step 2) extract visual features from the proposed regions and textual features from the natural language text, and finally (Step 3) rank candidate proposed regions using a metric that reflects the match between each region's visual features and the instruction's textual features. Experimental studies showed that REC performance was crucially dependent on the effective modeling of visual context, at scales corresponding to either the entire image [45], individual objects [122] or at multiple levels [121]. To reduce the high computational complexity and latency associated with the execution of three distinct stages, more recent REC approaches [129, 116, 117] have adopted a single stage approach with multi-modal fusion. These approaches employ multi-modal attention mechanisms that fuse verbal and visual cues, and often provide better modeling of contextual information at both global and local scales.

Despite the reduced complexity, such single-stage models for REC are still resource intensive and ill-suited to support real-time, interactive applications on resource-constrained embedded platforms, such as wearables and IoT devices. To support such real-time pervasive REC execution, we thus explore the use of neural optimization techniques that bypass redundant/inconsequential computation blocks in such single-stage models. Broadly speaking, such neural optimization can broadly involve either (a) static pruning, which reduces the model size by eliminating neural nodes or layers during the offline training phase, or (b) dynamic pruning techniques, such as convolutional layer sparsification [19] or dynamic routing [104], which selectively eliminate some of the computations in the complex backbone network during the *inference phase*. While dynamic pruning approaches are more nimble as they customize the computation to each input sample, all extant methods have been designed for uni-modal tasks (e.g., solely based on visual input). Our main contribution is to develop a novel, generalized dynamic pruning strategy for REC tasks, which are inherently *multi-modal* (consisting of both verbal and visual inputs).

This approach adopts the same basic principle associated with all run-time pruning methods:

skip computational blocks on a per-input basis, thereby reducing latency and computational energy overheads. However, our runtime pruning strategy explicitly uses the REC-specific property that the textual input contains critical information in identifying important/relevant regions in the image, and consequently develops mechanisms that optimize the visual processing pipeline (and the subsequent stages, such as attentional modules, that fuse visual and verbal cues) based on features embedded in the textual input. To the best of our knowledge, our work is the first to propose a multi-modal pruning approach for REC tasks. More specifically, we hypothesize that computational blocks at certain visual scale can be safely skipped depending on the sizes of both the target object and the objects referred to in the verbal input. Accordingly, in our approach, we use the textual features as a pivot to determine the necessity or relative importance of computing features at certain image scales. This approach contrasts with traditional unimodal runtime skipping mechanisms, where computational blocks are skipped primarily based on background vs. foreground differentiation and without regard to the size or saliency of individual objects.

Determining these scales is, however, a challenging problem for REC tasks due to the possibility of multiple relevant visual scales and saliency, such as when the textual reference is made with respect to another anchor object–e.g.,"small clock on the table", where the table and clock require different scales. Accordingly, adopting the prior *binarized* approaches (where a specific computational block is either executed in its entirety or completely skipped) runs the risk of missing crucial contextual information. Based on empirical observations that corroborate this anticipated pitfall of binarized skipping, we thus introduce a novel *"soft-skipping"* strategy, where certain computational blocks determined to be suitable for skipping are approximated using an alternate (convolutional) pathway that consumes dramatically lower computational resources. We believe that this approach, called **SoftSkip**, represents a general and powerful design paradigm for optimizing neural computation for tasks, such as REC, that involve correlated multi-modal inputs and require processing at different visual scales. We design a modified single-stage REC model, called *LGMDP* (Language-Guided Multi-modal Dynamic Pruning), that incorporates the SoftSkip mechanism into multiple stages, such as visual feature extraction, adaptive feature selection and global attention computation, associated with the ex-

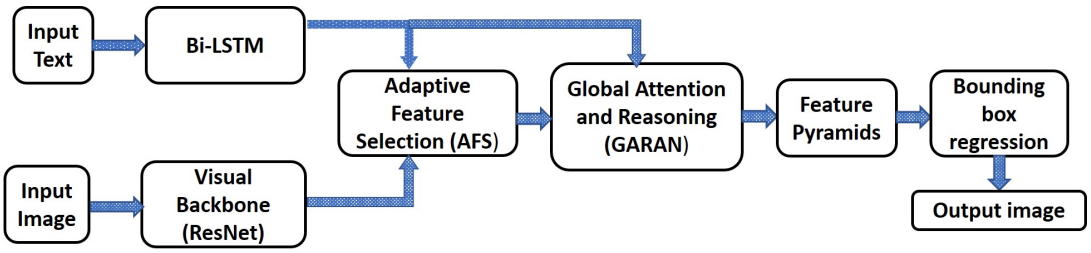


Figure 4.1: RealGIN single-stage referring expression comprehension model

ecution of REC tasks. With the proposed LGMDP model, we demonstrated the feasibility of dynamic model optimization for human instruction understanding to support low power, low latency execution on a pervasive device. Via extensive studies, we demonstrate how LGMDP provides significantly superior performance compared to standard static and dynamic pruning approaches, achieving lower latency while offering far higher comprehension accuracy (almost comparable to a non-optimized heavyweight baseline model).

Overall, we believe that LGDMP's paradigm of language-driven SoftSkip-based dynamic pruning represents a significant, foundational advance towards the goal of supporting accurate, real-time REC on embedded and pervasive devices.

4.1.1 Baseline Model

We use the RealGIN model [129] as the representative, state-of-the-art single-stage model for real-time, multi-modal REC tasks. As shown in Figure 4.1, RealGIN employs a ResNet [43] based backbone for extracting features from the image and a bi-directional LSTM [39] for extracting language features from the verbal instruction. RealGIN then uses a novel adaptive feature selection (AFS) module that identifies image features that are relevant to the text instruction. This is followed by a new multi-modal attention mechanism (GARAN) to facilitate language-guided visual attention. As shown in [129], RealGIN's achieves a 10x improvement in throughput, while achieving accuracy very close to that of multi-stage approaches such as MAttNet [121].

4.2 LGMDP

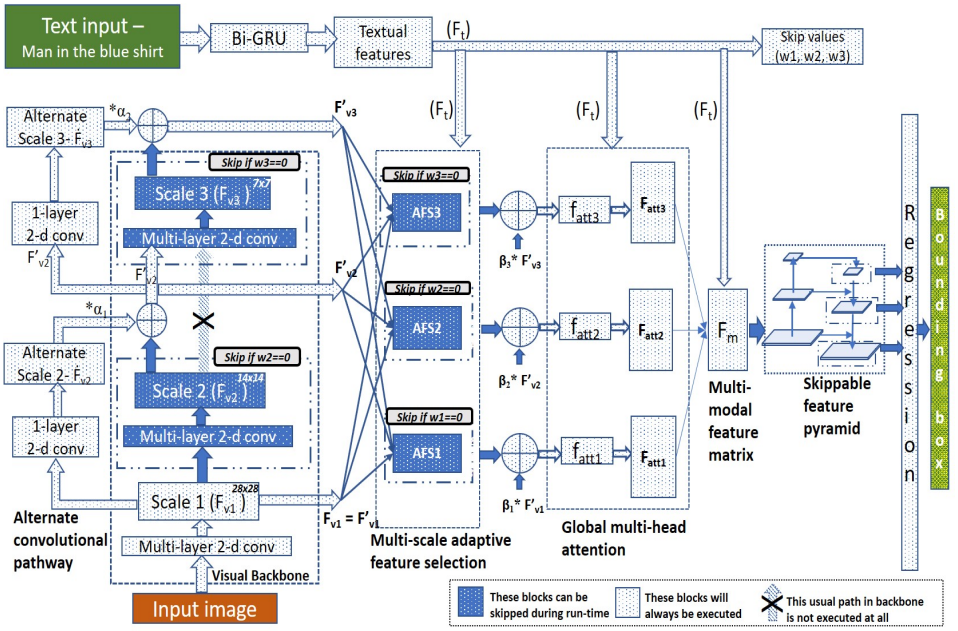


Figure 4.2: LGMDP - Model architecture

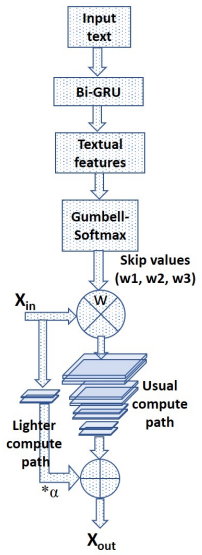


Figure 4.3: Soft-Skipping

Figure 4.2 presents the architecture of our proposed LGMDP model for single-stage, dynamically optimized REC. The model includes (a) a novel skippable visual backbone which enables input-specific, alternate efficient pathways for visual feature extraction, (b) a bi-directional GRU for textual feature extraction, (c) a module to compute multiple scale-specific skip factor, (d) a modified and skippable multi-scale adaptive feature selection (borrowed from RealGIN), (e) a global attention module (borrowed from RealGIN), (f) a skippable feature pyramid network and finally (g) bounding box regression layers (similar to YoLo3). The parameters of all these modules are trained using an end-to-end supervised learning process.

Our run-time SoftSkip approach (shown in Fig 4.3) is based on the following set of principles:

(a) The information embedded in the textual input should be used to determine which scales in the visual backbone are relevant. This choice of verbal→visual dependence is modeled on known models of human comprehension [21, 55], which show that humans utilize verbal cues to adjust visual attention (and not vice versa) and is driven by the high complexity of the visual feature extraction process.

(b) Since it is difficult to precisely determine whether the skipped layers of the visual back-

bone contain pertinent contextual information, we do not completely eliminate a computational block but instead use an alternative computationally-lightweight computational pipeline to approximate the features derived by the block. The approximated features continue to serve as useful input for the subsequent processing blocks.

(c) Processing blocks in the subsequent AFS and FPN stages, which fuse verbal and visual features, utilize the same set of scale-specific skipping parameters used by the visual backbone. In other words, a single set of skipping parameters are used across multiple processing modules, implying that if a particular NxN scale is skipped (or approximated) in the visual backbone, the corresponding NxN scale is also approximated in the subsequent AFS and FPN modules. Using a common set of 3 universally-applied skipping parameters also avoids unnecessary additional computation, as compared to an alternative approach where different stages of the model are associated with different sets of skipping parameters.

In this work, we compute 3 distinct skipping parameters, which are applied to conditionally skip the corresponding last three stages of the visual pipeline. The choice of 3 parameters, corresponding to scales of 28x28, 14x14 and 7x7, respectively, are chosen as they intuitively correspond to small, medium and large-sized objects, respectively.

In the training mode, all the blocks are still computed and multiplied by the skipping parameter. This way we make sure the model weights are differentiable for backward propagation. Formally, let C be the computational block to be skipped, x be the input to the computational block, w be the skipping parameter, $C^{'}$ be the low complexity approximation to the computational block, α be a trainable scaling parameter and x_{out} be the output. Then, forward computation in training mode is defined as follows.

$$x_{out} = w * C(x) + \alpha * C^{'}(x); \quad where\ w \in 0,1,\ \alpha \in [0,1] \tag{4.1}$$

During run-time, in order to achieve true latency savings we propose a different forward propagation where the decision to execute a computational block or not is based on the value of the

relevant skipping parameter. When the skipping parameter is 0, relevant computational block will not be executed returning latency savings. In general, let C be the computational block to be skipped, x be the input to the computational block, w be the pruning weight, $C^{'}$ be the low complexity approximation to the computational block, α be a trainable scaling parameter and x_{out} be the output. Then, forward computation in inference mode is defined as follows.

$$x_{out} = \begin{cases} C(x) + C^{'}(x) \ if w = 1 \\ \alpha * C^{'}(x) \ if w = 0 \end{cases} \tag{4.2}$$

4.2.1 Language-based scale-specific skipping parameters

We use the Gumbell-Softmax activation function on the language embedding generated by the GRU network to compute 3 discrete values that serve as common skipping parameters across different stages of the overall neural model. The language features (f_t) are captured by a 256-dimensional vector. This embedding vector serves as an input to a fully-connected layer with 3 output neurons. We then use the Gumbell-Softmax activation mechanism to obtain the 3 corresponding discrete binary values as follows.

$$w_1, w_2, w_3 = G(F(f_t)); \quad where \ w_1, w_2, w_3 \in \{0, 1\} \tag{4.3}$$

In the above equation, G is the Gumbell-Softmax activation function and F is the Fully-connected layer function. A value of '0' for a skipping parameter implies that the relevant module will not be executed (or, more precisely, will only be approximately executed), while a value of '1' implies normal execution of the module. In our current design, for a given input referring expression, only one of these parameters are assigned a value of '1', while the others are set to '0'. This discrete value of either '1' or '0' is taken through the output of Gumbell-Softmax activation function which directly returns a one-hot encoded 3-dim vector. Each dimension of this vector refers to a skipping parameter.

4.2.2 Skippable visual backbone

We evaluated two visual backbones for LGDMP, viz. (i) ResNet [129] and (ii) ShuffleNet [71]. Our baseline RealGIN model also uses the same ResNet visual backbone. A Resnet-based backbone is, however, not amenable to execution on low-resource embedded devices, such as the Jetson Nano. Therefore, we also consider lower complexity ShuffleNet as the backbone, to support real-time, on-device execution despite a loss in accuracy. ResNet has a residual structure and several earlier studies have proposed ways of skipping convolutions, using strategies such as early exit [99] and gated skipping [104], to reduce latency. However, such hard skipping strategies simply terminate computation of visual features at lower scales and thus cannot extract the multi-scale contextual information required for effective REC. In contrast, as shown in Figure 4.3, we adopt the SoftSkip approach: in LGMDP, whenever a certain block of convolutions at a certain scale is identified as suitable for 'skipping' in the visual backbone, a single-layer convolution is always used as an alternate pathway to provide approximate information at this scale. Accordingly, if F_{v1}, F_{v2}, F_{v3} denote the 3 visual feature scales in the backbone, we compute $\dot{F}_{v1}$, $\dot{F}_{v2}$ and $\dot{F}_{v3}$ which are ideally low complexity alternatives for F_{v1}, F_{v2} and F_{v3}. Then, we use w_1, w_2 and w_3 for skipping F_{v1}, F_{v2} and F_{v3} as follows.

$$F_{vi}^{'} = w_(i) * L_b(F_{vi}) + \alpha_i * Conv2D(F_{v(i-1)}), \quad for\ i \in 1,2,3 \tag{4.4}$$

In the above equation, L_b represents the usual convolutional blocks in the backbone which computes the respective feature scales. $Conv2D$ is a single convolutional layer with a stride of 2 and kernel size of 1. In the event that the w_i is 0, the corresponding feature scale will be approximated with this single convolutional layer.

4.2.3 Skippable Adaptive Feature Selection

The outputs from the different scales of the visual backbone are used as input to adaptive feature selection (AFS) modules (described in [129]) operating at the corresponding scales. The

function of the AFS module is to take the visual feature maps from different scales and project them to the same resolution and depth using multiple convolutional layers. The visual feature maps from different scales represent visual information at different semantic levels. Separately, the language priors for these semantic levels are learnt from the textual feature vector (f_t) resulting in three separate fusion weights (v_1, v_2 and v_3) corresponding to these feature scales. These language priors are found to be useful to represent the wide variations in the content of textual expressions. For example, when an expression contains semantic information such as color/texture, AFS can increase the fusion weights for low/mid-level features. After that, a final visual feature map is computed as the weighted sum of these individual feature maps.

In LGMDP, we use multi-scale AFS wherein three AFS modules are involved each producing output at three different scales. We then use the skipping parameters w_1, w_2 and w_3 calculated earlier to determine whether the AFS modules at a certain scale should be subjected to soft-skipping or not. By sticking to our principle of soft-skipping, whenever an AFS module is subjected to soft-skipping, its output is always approximated by a proportion of the corresponding visual feature map obtained from the skippable visual backbone (as described earlier). Formally,

$$x_i = w_i * AFS_i(F_{v1}^{'}, F_{v2}^{'}, F_{v3}^{'}, F_t) + \beta_i * F_{vi}^{'} \tag{4.5}$$

When w_i is 0, respective AFS stage output will be 0 and relevant AFS activated features will be approximated with $\beta_i * F_{vi}^{'}$.

4.2.4 Global Attention Module

In this module, the AFS-derived feature maps are used as input to an attention mechanism (mimicking that used in [129]). This module, called GARAN, uses the textual features to collect expression-related information over the whole image and then selectively diffuse this information to all anchors (predefined (location, size) templates for objects). As described in [129], the differential attention maps can be obtained by using the ground-truth bounding box of the target object, during training, as a supervision signal to calculate the attention loss.

Even though scale-specific skipping behavior can conceptually also be incorporated in the GARAN, we empirically found that such skipping resulted in a significant drop in accuracy with only minimal latency benefits. As the GARAN module is relatively computationally lightweight, there is not much benefit in implementing the skipping behavior at this stage. The output of the GARAN module is the multi-modal feature matrix F_m, which is then utilized to perform a YoLo-style bounding box regression.

4.2.5 Skippable Feature Pyramids and bounding box regression

The multi-modal feature matrix obtained from the GARAN module is used by a feature pyramid network [62] to perform bounding box regression. The feature pyramid network (FPN) is generally used to overcome the problem of limited receptive field exhibited by convolutional layers. FPN usually has two pathways: (1) *Bottom-up pathway*, which ideally is the feed-forward computation of convolutional layers with a scaling factor of 2, and (2) *Top-down pathway*, which up-samples feature vectors that are spatially coarser but semantically stronger, using a scaling step of 2. We apply our soft-skipping approach only to the top-down pathway in a similar fashion as described in the visual backbone. The FPN outputs feature maps outputs features at multiple scales that would be used for regression. Our intuition is that the language may have a hint on the size of the object of interest which may in turn help us to skip the irrelevant feature scales.

4.3 Results

We evaluate LGDMP's performance for REC tasks using three benchmark datasets.

1. **ReferIt or RefCLEF** [52] - This dataset contains about 19997 images selected from the ImageCLEF competition [41], with the objects in them referred through a two-player game, where the players alternate between generating and comprehending referring ex-

pressions. Apart from collecting this pioneering set of data of ¡*image*,*expression*¿ pairs, the authors also provided an analysis of the visuo-linguistic characteristics observed, such as the relationship between target bounding-box area versus the key-words used in the expressions etc. Thus, we used this dataset first to evaluate multiple SoftSkip strategies and empirically identify the preferred LGMDP model.

2. **RefCOCO** [122] - This dataset was also collected using the same game paradigm as ReferIt dataset, but the stimulus images are selected from the MSCOCO [63] dataset which is often used for training deep learning based object detectors. In the dataset, there are 142,209 expressions referring at 50,000 objects in 19,994 images.

3. **COPS-Ref** [29] - This is a recently released dataset containing 148,712 expressions, that collectively refer to 1,307,885 regions on 75,299 images, making it the current largest real-world image dataset for referring expressions. The ¡*image*,*expression*¿ pairs in this dataset are considered challenging due to the presence of "distractor" objects which are similar to the target objects. Thus, this dataset serves to analyze the deeper reasoning abilities, such as logic and relational inference, of various REC techniques.

To evaluate the inference latency (time taken to process a single <*image*,*expression*> pair) of LGDMP and other competing models, we implement and deploy these models on an *NVIDIA Jetson TX2* [7], a representative embedded device. TX2 comprises of a 256-core GPU,a dual-core NVIDIA processor and 8GB system memory.

4.3.1 Different SoftSkip and HardSkip Strategies

	ReferIt		
	Val	**Test**	**Lat(ms)**
LGMDP-skip1	68.21	65.17	290
LGMDP-skip2	67.90	64.98	255
LGMDP-skip3	63.16	60.18	200
LGMDP	67.19	64.56	220
LGMDP-Hardskip	57.54	53.11	218

Table 4.1: Comparisons of LGDMP variants with different skipping behavior settings

We first start by evaluating, on the ReferIt dataset, four distinct variants of our proposed SoftSkip strategy vs. a candidate Hardskip alternative, to help establish the preferred LGMDP alternative and its absolute performance. We evaluated a few different variants of SoftSkip behavior, resulting in 5 different LGDMP variants, as follows:

(a) *LGMDP-skip1:* In this variant, only the largest scale (scale 3=7x7) is amenable to dynamic runtime soft-skipping, with SoftSkip enabled across all of the backbone, AFS and FPN stages; the other scales are always computed in their entirety.

(b) *LGMDP-skip2:* In this variant, SoftSkip is enabled for two scales (7x7 and 14x14), across all of the backbone, AFS and FPN stages.

(c) *LGMDP-skip3:* Here, SoftSkip is enabled across all three scales (7x7, 14x14 and 28x28), and throughout the entire DNN including the backbone, AFS and FPN stages.

(d) *LGMDP:* In this preferred model (which diverges only minutely from LGMDP-skip3), SoftSkip is enabled for all 3 scales in the AFS and FPN stages, but it is applied only to the two larger scales (7x7 and 14x14) on the visual backbone.

(e) *LGDMP-HardSkip:* This variant is identical to LGMDP, except that it replaces SoftSkip with a hard skipping strategy, where the computation for DNN states identified for skipping is eliminated entirely (instead of computing an approximate set of features).

Table 4.1 compares their relative performance. As expected, the average accuracy values degrade slightly as more scales are progressively possible candidates for soft-skipping; conversely, the overall execution latency decreases as well. We observe that option (d) (where skipping scale 1 in the visual backbone is prohibited) performs significantly better (suffering an accuracy drop of $< 1\%$) than LGMDP-skip3 (which suffers an accuracy drop of $\sim 5\%$), while exhibiting comparable latency. Accordingly, we pick and use (d) as our preferred LGMDP embodiment for all subsequent experiments.

We also note that LGMDP-Hard suffers a significant ($> 12\%$) drop in accuracy, validating our belief that completely eliminating feature extraction at certain scales is inadvisable for REC

tasks where language-guided visual reasoning often occurs at multiple scales. We hypothesize that this performance loss could be due to two different factors: (a) *Vanishing Gradient Problem:* During the learning phase, if the skipping weight equals 0, the relevant feature scale results in a null matrix, which results in a zero gradient that in turn affects the efficacy of backpropagation; (b) *Over-reliance on language features:* Hard-skipping implicitly assumes that the verbal instruction provides sufficient cues for determining the appropriate visual scales needed. This is, of course, not universally true; in cases where the verbal cues are imprecise, hard skipping effectively obliterates features at certain scales, making the eventual recognition of target objects extremely difficult. SoftSkip, in contrast, is more permissive of situations where the language pruner makes mistakes.

4.3.2 Qualitative Insights on Multi-Scale SoftSkip

Before returning to using macroscopic metrics, such as comprehension accuracy, to compare LGMDP against non-dynamic baselines, we analyze the corpus of instructions in the ReferIt dataset to reveal deeper insights into the functioning of our proposed SoftSkip strategy.

Figure 4.4 provides some typical examples of the referring expressions and the associated images in the ReferIt corpus. The top row contains examples of images where the referred object is generally large in size. The first two images in the middle row contain references to small target objects, while third image contains a reference to a prominent target. The last row of images are examples of where the verbal expression employs color attributes. We can observe that the textual references in these examples often provide indicative hints about the most salient visual scale. Small objects may need a scale of 28x28 (scale 1), big objects may need a scale of 7x7 (scale 3), references to low-level image properties like color/texture may benefit from scale 1, while the presence of relative positional references (left/right/foreground/background) may suggest the need for multiple scales. A fairly detailed analysis of the visuo-linguistic characteristics of the ReferIt dataset was provided by the authors in [52]. We used this to examine the semantic relevance of the SoftSkip behavior exhibited by LGMDP. The individual bars in

Figure 4.4 provide the percentage of expressions where LGMDP chose a certain scale for full execution without skipping (with the corresponding skipping parameter $w_i == 1$), for instructions that contained specific keywords such as 'big', 'little', 'foreground', 'background' etc. In CNN-based object detection, spatial resolution diminishes rapidly as computation proceeds to the deeper layers; accordingly, deeper layers are likely to be less useful for capturing features of smaller objects. Therefore, one would expect scale 1 to be active and scale 3 skipped ($w_1 = 1, w_3 = 0$,) more often for scenarios involving detection of small objects. Similarly, one would expect scale 3 to be active ($w_3 = 1$) mainly for target images with larger bounding box area and expressions needing visual context corresponding to larger object sizes.

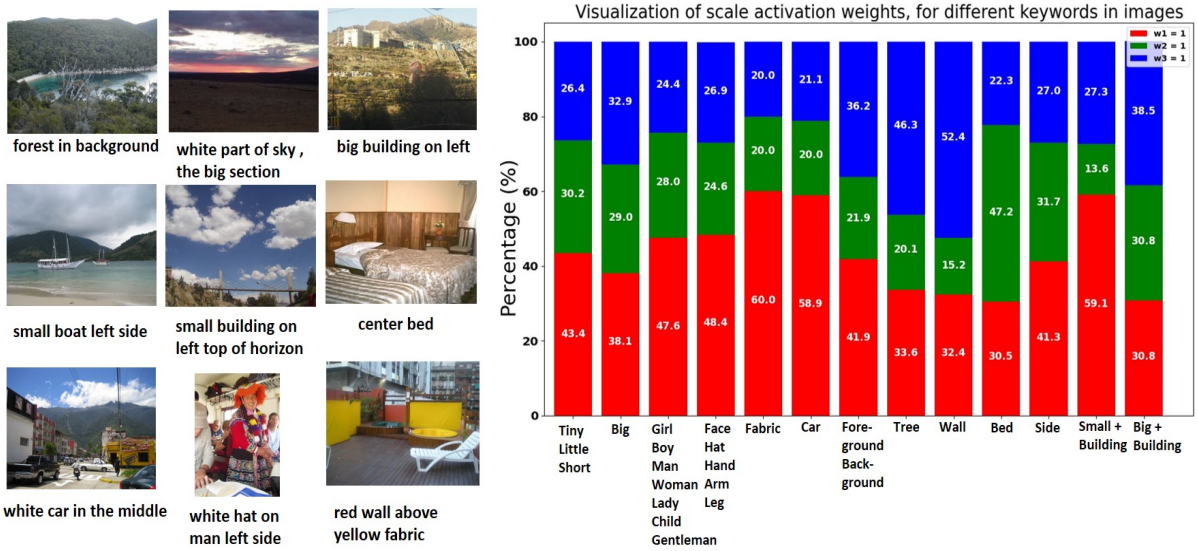


Figure 4.4: Examples of size, position and color based references in the ReferIt dataset and a visualization of scale activation weights, for different keywords in images

We additionally observe the following relevant points,

- As reported in [52], in the ReferIt dataset, the bounding box area of the target objects was found to be larger for expressions involving the keyword 'big', whereas the keywords 'tiny, little and short' were often associated with target objects with smaller bounding boxes. Correspondingly, in our skipping mechanism, the scale 1 (28x28) is used without skipping ($w_1 = 1$) for 43% of the instructions containing 'tiny, little and short', whereas the scale 3 (7x7) is used without skipping ($w_3 = 1$) for 26% of such instructions. When we looked at instructions containing the word 'big', the proportion of the instructions for $w1 = 1$ dropped to 38% while the proportion for scale 3 ($w_3 = 1$) increased to 32%

of the instructions. The results broadly confirm our hypothesis that larger target objects would preferentially activate coarse scales (scale 3) more often. Note, however, that additional contextual factors, beyond just the target's bounding box size, can affect the choice of scale, and in fact, require *multiple* scales. For example, consider the *ReferIt* expression: "the door of the big building". Here, while the target object is a 'small' door, comprehension requires identification of the 'big building' first.

- We then dug deeper to filter out cases where the references to the same object are made but with different size adjectives. We chose 'Building' as our target object and selected the expressions containing the word 'Building'. Then within these referring expressions, we looked at references to 'big' building and 'small' building separately. We found that in cases involving 'small' buildings, scale 1 was used for 59% of the expressions. In contrast, for expressions that involved 'big' building, scale 1 was used only about 30% of the time, while the use of scales 2 and 3 increased dramatically.

- In this dataset, color adjectives were used very often to refer to relatively smaller target objects, such as 'car' and 'fabric'. Accordingly, we observed that scale 1 was activated in LGMDP ($w_1 = 1$) for a vast majority of instructions involving these objects.

- Relative references in referring expressions may involve a mix of words such as positional keywords or objects. In ReferIt, the words 'foreground', 'background', 'side (left side, right side)', 'guy', 'man', 'woman', 'tree' and 'wall' were among the most frequently used to describe relative location of the target object. We observed a mix of scales being activated for these keywords. First, we looked at expressions that involved explicit references to foreground/background and side keywords. From a visual analysis in Figure 4.4, we could not determine an obvious 'preferred' scale associated with these words. Next, we looked at the expressions involving relative position with respect to the objects ('tree' and 'wall'). We observed that Scale 3 ($w3 = 1$) was fully activated for a large proportion of such expressions. However, for references to human objects (e.g., girl, boy, woman, man, child, lady, gentleman) or constituent body parts (e.g., face, hand, arm and leg), Scale 1 was activated more often. We note that human objects and/or their body

Method	Backbone	ReferIt			COPS-Ref		RefCOCO			
		Val (%)	Test (%)	Lat(ms)	Test (%)	Lat(ms)	Val (%)	Test A (%)	Test B (%)	Lat(ms)
RealGIN	ResNet-152	68.29	65.37	330	48.17	330	75.10	76.71	68.22	330
LGMDP	ResNet-152	67.19	64.56	220	46.81	225	73.27	75.19	67.23	218
RealGIN-SkipNet	ResNet-152	61.49	58.31	250	38.66	250	69.12	70.23	72.10	250
RealGIN-staticPr	Shufflenet	51.21	48.29	150	45.98	150	60.45	62.29	64.97	150
LGMDP-staticPr	Shufflenet	49.90	47.11	120	44.12	124	58.10	61.11	63.27	120

Table 4.2: Performance comparison of LGMDP against RealGIN, Static Pruning and Dynamic Pruning

parts are often easily described via the use of low-level adjectives (e.g., color/texture for skin or clothing).

- The authors in [52] had reported that the target object 'bed' was very often referred by 'absolute location' ("center bed", "bed on the left" etc.). In LGMDP, we find that medium scale 2 is invoked for a large proportion of these expressions.

These observations intuitively justify the scale-specific skipping strategy adopted in LGMDP, where a necessary scale is activated according to the semantic level of the visual context needed. Given that a single visual scale may not be directly inferred from verbal instructions or may not be appropriate for many images, LGDMP's SoftSkip approach helps cushion the effect of erroneous computation of skip weights.

4.3.3 Comparison of LGMDP with RealGIN

We now compare the performance of LGDMP with our chosen baseline - RealGIN. From Table 4.2, we observe that LGDMP's performance is very close to that of the baseline RealGIN approach. The drop in accuracy is a very meager $<1\%$ for ReferIt and RefCOCO (testB) datasets. The overall accuracy of RealGIN as well as LGMDP is lower for the more challenging COPS-Ref dataset, with LGMDP suffering a more discernible ($\sim$1.3%) relative loss in comprehension accuracy. Such modest loss in accuracy is, however, balanced by the significant ($\sim$33%) reduction in latency achieved by our dynamic pruning approach. The latency of LGDMP (220 msec, 225msec and 218msec for the ReferIt, COPS-Ref and RefCOCO dataset respectively) is about 105-110msec lower than the baseline RealGIN.

4.3.3.1 Pervasive versions: Static Pruning vs. Dynamic SoftSkip

The original RealGIN uses ResNet-152 as its visual backbone. However, ResNet-152 is computationally intensive and not suitable for pervasive applications–e.g., the original RealGIN model cannot be executed on the Jetson Nano, a more resource-constrained pervasive device. As a form of static pruning, RealGin's ResNet-152 visual backbone can be replaced Shufflenet [71], a lightweight DNN model specially curated for pervasive applications with low latency and memory requirements. We thus experimented both with (a) a static pruning model, *RealGIN-staticPr* (where Shufflenet is used as part of the RealGIN backbone), and (b) as well as *LGMDP-staticPr* (an exemplar of dynamic soft skipping, where our SoftSkip paradigm is applied on top of RealGIN-staticPr). Applying static pruning does result, as expected, in a significant (18%) loss of accuracy, while enabling a 2x faster execution on the Jetson Nano. In comparison, LGMDP suffers only a marginal 0.8% degradation in performance while enabling a 33% further reduction of latency (from 330 msec to 220 msec). Similar results hold also for execution on our representative Jetson TX2 (see Table 4.2), demonstrating the general applicability of our proposed SoftSkip approach.

4.3.4 Dynamic Pruning: SkipNets vs LGMDP

In contrast to existing dynamic pruning approaches designed for uni-modal neural models, LGMDP is designed for a multi-modal REC task. To evaluate the benefit of our multi-modal dynamic pruning strategy against the existing uni-modal pruning approaches, we take Skipnet [104] as an alternative baseline. Skipnet uses a gated network architecture where individual layers are dynamically skipped (or not), based on gating parameters computing using the Gumbell Softmax activation function applied on existing residual blocks in the convolutional backbone. For comparative assessment, we implement *RealGIN-SkipNet*, a model where we replace RealGIN's visual backbone with the Skipnet architecture. As shown in Table 4.2, RealGIN-Skipnet suffers an accuracy loss of $\sim$7%, in contrast to a mere 0.8% accuracy loss for LGMDP, while also incurring a $\sim$10% higher latency than LGMDP. This result conclusively establishes the

superiority of our proposed SoftSkip-based dynamic pruning for multi-modal REC tasks.

4.4 Discussion

While the dynamic optimization paradigm in SoftSkip has been applied to the problem of language and vision REC tasks, we note that LGMDP can be easily extended to accommodate pointing gestures by including a pointing affinity field in the GARAN mechanism. We also believe that calculating pointing affinity maps would also involve deep neural networks operating at multiple scales, while factoring in the aforementioned distance dependence of pointing fidelity. Thus, we believe that scale-specific soft-skipping could still be relevant in such situations.

Since the introduction of SoftSkip, there have been notable advancements in REC models, particularly with the emergence of transformer-based REC models like TransVG [32]. While these transformer-based models may lack scale-specific features, they typically incorporate multiple attentional layers stacked together. We posit that our SoftSkipping strategy could be effectively applied to selectively execute these attentional blocks, thus potentially reducing overall latency in transformer-based REC models.

4.4.1 Achieved Design Goals

In SoftSkip, we utilized features from the language pipeline to optimize the processing pipeline of the REC model. Through this dynamic model optimization, we achieved both design goals: (a) low latency sense-making and (b) reduced energy consumption that is more amenable to on-device execution on battery-powered devices. While all baseline models, including RealGIN, met the design requirement of inference latencies well below 1 second, our proposed LGMDP model further reduced inference latency by approximately $\sim 33\%$, with only a $\sim 1\%$ reduction in task accuracy. This improvement translates to a similar increase in operating time on battery-

powered devices. This is a significant enhancement, considering that continuous execution of such existing comprehension models on smart glasses or robotic platforms would typically drain their batteries within 2-3 hours.

However, since SoftSkip does not inherently support pointing gestures, the focus of the later chapters will be on adding support for these additional modalities. This will help achieve the first design goal of multi-modal sense-making while maintaining reduced latency and energy consumption.

4.5 Summary

In summary, SoftSkip and the proposed LGMDP model demonstrated the possibility of dynamic model optimizations to achieve low power and low latency execution of REC models on a pervasive device. This work makes the following key contributions:

- We introduce a novel run-time DNN optimization approach called LGMDP that is useful for supporting multi-modal tasks such as REC. To the best of our knowledge, LGMDP is the first model that uses textual features as a pivot to skip computations in both the visual processing and the subsequent multi-modal fusion stages. In addition, LGMDP employs the novel concept of SoftSkip, where computational blocks are not completely eliminated but rapidly approximated, thereby ensuring that features at different visual scales are at least partially preserved.
- We implement LGMDP, as well as a variety of competitive state-of-the-art (SOTA) alternatives. Using three different benchmark datasets (ReferIt[52], RefCOCO [122] and Cops-Ref [29]), we show that LGMDP offers a far superior accuracy-vs.-latency trade-off and is able to offer a significant reduction in computational latency with negligible loss in accuracy. In particular, LGMDP suffers only an $\sim$0.5% loss (65.3%$\rightarrow$64.6%) in comprehension accuracy compared to the non-optimized RealGIN baseline, but achieves more than 33% reduction in processing latency when executed on a NVIDIA Jetson TX2

device. In addition, LGMDP's accuracy of 65.37% @ 220 ms of latency far outperforms the best performing SOTA uni-modal pruning alternative (only 58.31% accuracy at similar latency). In addition, in Section 4.3.2, we provide more granular insights into how LGMDP's multi-scale SoftSkip mechanism is able to leverage on appropriate textual cues.

- We also demonstrate how LGMDP's SoftSkip-based approach can be combined with standard static pruning approaches to support ultra-lightweight, real-time REC execution on the Jetson TX2, a representative embedded platform. While the combined model suffers an 18% loss in accuracy compared to the RealGIN baseline, it achieves a significant 2.75x reduction in latency and 7x reduction in memory overhead, which is superior to that achieved by static pruning alone (17% accuracy loss with 2.2x and 7x reduction in latency and memory, respectively).

Chapter 5

COSM2IC: Optimizing Real-time Multi-Modal Instruction Comprehension on Pervasive Devices

In this chapter, we introduce the paradigm of dynamic model optimization into Vision, Language, and Gesture-based REC models to facilitate real-time, on-device execution of multi-modal instruction comprehension. In contrast to the previously studied SoftSkip technique, this approach incorporates pointing gestures as an additional modality to enhance the interactivity in human instruction understanding. Although we explored the effectiveness of pointing gestures in Chapter 3 using M2Gestic, the evaluation primarily took place in a synthetic setup. In this chapter, our objective is to introduce and assess a more realistic dataset, while preserving the object clutter and ambiguity aspects present in M2Gestic.

Firstly, we present a newly curated multi-modal instruction dataset that showcases a human-robot collaborative tabletop target acquisition task. This dataset differs from previous ones [87, 52, 56] by providing a more realistic corpus collected with simultaneously issued verbal and gestural instructions.

Subsequently, we introduce our optimization framework, termed COSM2IC [107], and present

the results of accuracy, latency, and energy consumption on the collected dataset.

5.1 Introduction

Supporting the comprehension of multi-modal target acquisition instructions is driven by increasingly sophisticated *coupled* Deep Neural Network (DNN) models such as [121, 72, 122], where initial mode-specific stages are followed by additional layers that combine cross-modal information. However, when applying these models to the two scenarios mentioned in section 1.2, we encounter the following challenges.

- DNN models are typically trained and evaluated on platforms that are capable of supporting their high computational and energy needs to achieve maximum accuracy. However, applications such as the shopping virtual assistant mentioned in 1.2, require *on-device execution* of models in highly resource constrained environments (Microsoft HoloLens or Nvidia Jetson platforms), where it is infeasible to support their high computational and energy needs. On the other hand, reducing the complexity of these models via optimization techniques leads to lower task performance. Hence, there is a simultaneous need to minimize such loss in accuracy.

- Model optimization strategies such as compression [119] and convolutional layer sparsification [19] have primarily tackled *single-modality* tasks, such as conversational assistance (e.g., [23]) and object recognition (e.g., [17]), whereas multi-modal instructions often exhibit significant contextual variations in the amount of information conveyed by a specific modality. For example, a user may offer more elaborate verbal cues, such as 'the hammer next to the yellow gear shafts behind the hydraulic jack', when indicating a specific object in a highly cluttered environment, but prefer a combination of pointing+ shorter verbal commands (e.g., 'that red hammer') when dealing with a less-cluttered scene. A single optimized model is unlikely to be able to *adapt* to such dynamic variations in the complexity of individual modes.

- As we shall show in Section 5.6.3, due to an expanded set of cross-modal feature embeddings and attention mechanisms needed for multi-modal instructions, approaches such as model compression and sparsification perform poorly when faced with complex, multi-modal coupling of features. Therefore, beyond a certain level of optimization, the performance of optimized models drop-off drastically.

- The DNN-based comprehension models in the prior work have all been trained on datasets that are acquired from a third-person viewpoint, whereas our scenario Figure 6.1 would involve understanding the human instruction from a first-person (human instructor's) viewpoint. This comes with well-known challenges due to factors such as the motion of the head-mounted camera, occlusion etc. as well as perspective ambiguity. Furthermore, these models deal only with images/videos without the depth-based gesture information, making them less applicable to scenarios like our use case depicted in Figure 6.1. Recent studies [87, 106] show that the performance of instruction comprehension is significantly affected by object clutter, view-point ambiguity and also distance from which pointing is performed.

5.2 Multi-modal instruction corpus

Existing DNN-based comprehension models have all been trained on datasets acquired from third-person viewpoints. However, natural human instruction understanding should ideally occur from a first-person perspective, also known as ego-centric. Enabling instruction comprehension from a first-person viewpoint poses challenges due to factors such as head-mounted camera motion, occlusion, and perspective ambiguity. Moreover, these models typically only handle images or videos without incorporating depth-based gesture information. To address these limitations, we introduce an ego-centric multi-modal dataset for target acquisition instructions. To build our corpus of ego-centric multi-modal target acquisition instructions (*simultaneously* utilizing voice, vision, and gesture), a common table-top-based target selection task was chosen as a canonical use-case that can be extended to several applications. For such

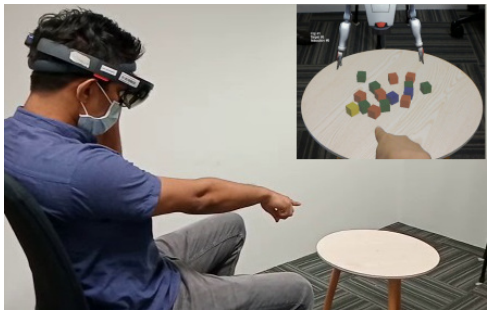


Figure 5.1: User Study-1: Blocks and a robot viewed via HoloLens.

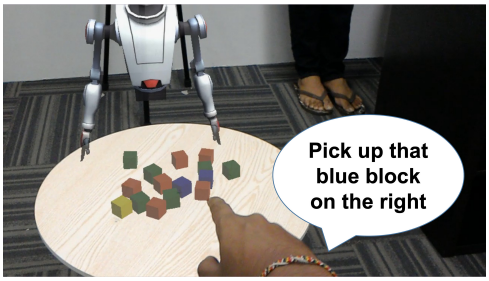


Figure 5.2: Cluttered Blocks-CB

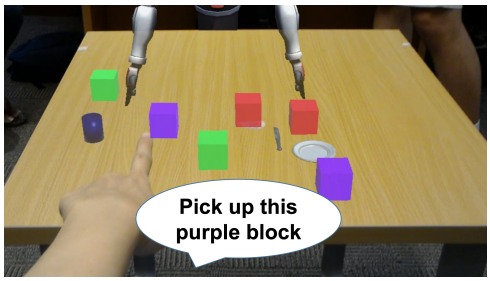


Figure 5.3: Uncluttered Blocks-UB

Figure 5.4: Cluttered Realistic-CR

Figure 5.5: Uncluttered Realistic-UR

Figure 5.6: HoloLens-assisted collection of multi-modal instructions in Study-1 setup.

a task, prior studies in HRI literature viz., *Collaborative Manipulation Corpus (CoMC)* [87] and the *Embodied Multi-modal Referring Expressions (EMRE)*[56] have carefully designed the table-top setups to emulate different levels of clutter and elicit human instructions with varying level of ambiguity but without the support of concurrent gestures. Hence, we draw upon these studies to curate our corpus. Accordingly, the COSM2IC dataset (approved by the IRB of Singapore Management University) consists of four types of setups used to elicit referring expressions from human subjects.

1. Cluttered blocks (**CB**): Contains 15 colored blocks (Fig 5.2) arranged using the 14 unique block arrangements provided in CoMC.

2. Uncluttered blocks (**UB**): Contains 6 blocks with additional real-world objects (Fig 5.3). We selected 50 block arrangements from the EMRE dataset.

3. Cluttered realistic objects (**CR**): Incorporates 15 colored containers present in a Mixed Reality (MR) kitchen arrangement along with several other realistic virtual kitchen objects. (Fig 5.4). The containers were arranged using the 14 unique block arrangements in CoMC.

4. Uncluttered realistic objects (**UR**): Incorporates 6 colored containers together with realistic virtual objects (Fig 5.5). The container locations were arranged similarly to the selected 50 block arrangements from the EMRE dataset.

5.2.1 User Study Setup

We conducted two user studies (a) to elicit a set of natural table-top manipulation instructions, and (b) to establish a performance baseline on how a human instructee would comprehend such instructions.

5.2.1.1 Study 1–Instruction Corpus

Figure 5.1 illustrates the high-level setup for Study 1. A human instructor, wearing a HoloLens Mixed-Reality headset presents a set of *virtual* table-top objects, overlaid on the physical table. The objects are only visible via the headset. A custom HoloLens application presents each instructor with a series of object arrangements (randomly chosen from the 50 uncluttered and 14 cluttered setups), with a red arrow indicating the designated target object. For each of the 14 cluttered (CoMC) table-top arrangements, we randomly chose 8 distinct blocks as the target blocks; for each of the 50 uncluttered (EMRE) arrangements, we chose 2 target objects, resulting in a total of 112 distinct cluttered and 100 uncluttered setups.

The participant then instructs the virtual robot to pick up the designated target block. Participants were free to use any words (e.g., 'pick up the green block on the left, next to two yellow blocks'), as well as any (or no) pointing gestures. The HoloLens application recorded the audio, camera (RGB), and depth sensor data associated with the issued instruction from

the instructor's Point-of-View. The participants were free to take breaks and completing all the instructions was not compulsory. Each data collection session lasted $\sim$ 1 to 1.5 hours and involved the capture of $\sim$ 100 distinct instructions. Some participants performed multiple such sessions on different days. For the UB and CB parts, Study 1 was performed by a total of 28 distinct subjects (19 male), with their ages having mean = 25.14 and s.d.= 1.96. We manually (a) transcribed the oral commands to create a corpus of text instructions, and (b) drew a bounding box around the target object in the RGB image frame. For the pointing gesture, we assumed that the ground truth pointed location to be the center point of this drawn bounding box. During the annotation process, we also removed certain samples with device recording errors and erroneous data points. Thus, we ended with 2566 instructions for UB and CB setups. The same procedure was used to obtain 510 instructions for UR and CR setups. However, only a smaller number of participants (6) were involved in generating instructions for the UR and CR-based setups. In general, the word length of the instructions elicited was lower than the corresponding values reported in the CoMC and EMRE corpus. This is possibly due to the simultaneous use of pointing gestures along with verbal instructions, and arguably demonstrates the greater ecological validity of our experimental setup.

5.2.1.2 Study 2–Baselining Human Comprehension

Study 2 shows the performance of human subjects in interpreting the multi-modal instructions collected from Study 1. Amazon Mechanical Turk [15] was used to recruit 487 participants who were shown an image frame (containing the table-top arrangement and pointing gesture) along with the verbal instruction. Each participant was tasked with identifying and placing a red dot on top of the target object. For each instruction, we targeted obtaining annotations from three unique participants. We rejected invalid responses via several checks (e.g- if the participant had taken $\leq$ 5s to provide a response or if the overall accuracy of a participant is $\leq$ 20%). Among a total of 2566 block-world based instructions (UB and CB), we received 3 valid responses for only 1349 of them. These filtered 1349 instructions were then used to evaluate a baseline of human performance in understanding the instructions. However, for the evaluation

No	**Question**
1	**Ambiguity:** Could any other object(s) also be chosen for the multi-modal instruction? (a) No (b) Yes - one other object (c) Yes - multiple
	For the following questions (2–5), choose a rating for "easiness" on a scale 0 - 4; 0 → Impossible, 4 →Very Easy
2	**Easiness- task:** Was it easy to accurately identify the target?
3	**Easiness- color:** Was it easy to accurately identify the color of the target object?
4	**Easiness- gesture** Was it easy to accurately identify the objects that are NOT near the pointed location?
5	**Easiness- Spatial relations** Was it easy to accurately identify the objects that DONOT satisfy the spatial relations in the verbal instruction?
	For the following questions (2–4), choose a rating for "importance" on a scale 0 - 4; 0→ Not needed/Unavailable, 1→ Good to have, 2→Necessary, 3→ Important, 4→Crucial
2	**Importance- color:** To accomplish this task, how important was the color of the target object?
3	**Importance- gesture:** To accomplish this task, how important was the pointing gesture?
4	**Importance- spatial relations:** To accomplish this task, how important were the spatial relations mentioned?
5	**Clarification:** Any single clarification question you want to ask? *Free-text,"None" accepted.*

Table 5.1: Key Survey Questions for Each Study 2 Instruction

of machine learning models, we did not restrict ourselves to these 1349 instructions. Similarly, for 510 instructions from CR and UR parts using the virtual kitchen environment, we received valid responses for 368 instructions.

Besides annotating their choice of object, each subject answered a series of questions (see Table 5.1 for details) eliciting their perceptual response to *each* such interpreted instruction. Since completing the survey for every instruction could induce fatigue, we restricted the survey to only the block-world based instructions. In spite of this, several participants did not provide complete responses to all the survey questions. As a result, within the 1349 valid responses to block-world instructions, we further selected a subset of 1025 of the instructions for which there were three valid survey responses to infer some key insights (presented below).

5.2.2 Insights from Study 2 (Human Comprehension Baseline)

From study 2, we analysed the accuracy of human responses to the instructions as well as their perceptions about ambiguity in the instructions and the easiness of the task. These results are

Table 5.2: Summary of survey results

No. Of correct responses	% age of inst	**Words**	**Importance** (Avg. rating) **Color**	**Gest**	**Spatial**	**Ambiguity** **Yes** (% of inst)	**No**
0	16.1	7.44	2.71	2.16	2.41	12.6	3.5
1	27.4	7.87	2.88	2.43	2.60	17.9	9.6
2	33.9	8.94	3.10	2.48	2.92	14.6	19.2
3	22.6	9.16	3.36	2.45	3.15	4.6	18.0
Total	100	–	–	–	–	49.7	50.3

summarised in Table 5.2. The instructions were first categorized according to the number of correct human responses (in the range of 0–3), as indicated by the first column of Table 5.2. Then, for each of these categories, Table 5.2 summarizes the word-length of verbal instructions, importance ratings for the three types of information in the instructions and the perceived ambiguity in the instruction. Overall, for about 16% of the instructions, none of the participants were able to pick the right block.

Perceived easiness of task: A rating of 0 or 1 is considered *difficult*, a rating of 2 is considered *normal*, 3 or 4 is considered *easy*. For the cases where all three responses were right, an overwhelming proportion of the participants found it easy. For about 16% of the instructions, none of the participants were able to pick the right block. In general, as anticipated, the number of accurate responses received was inversely related to the perceived difficulty level of the instructions (lower difficulty $\longleftrightarrow$ higher response accuracy). However, subjects often provided high easiness rating even though they made errors and found some of the instructions ambiguous.

Ambiguity: A total of 49.7% of the instructions were perceived to be ambiguous by the participants of study 2. As expected, instruction ambiguity and response accuracy were inversely related: the proportion of instructions labelled as ambiguous was significantly low for the instructions where all three participants successfully identified the target, and vice-versa. However, in spite of the reported ambiguity, the subjects were able to accurately identify the correct target in 74.17% of the ambiguous instructions.

Word-length: The average word-length of the verbal instructions (collected by Study 1) was observed to be generally higher for those instances where a greater fraction of humans were able

to successfully identify the target block. In general, the wordlength of the instructions elicited was lower than the corresponding values for the CoMC and EMRE corpus. This is possibly due to the simultaneous use of pointing gestures along with verbal instructions, and arguably demonstrates the greater ecological validity of our experimental setup. There are three main types of information provided in the instruction, viz. the color of the target object, the pointed location and the spatial relations with respect to other objects. The color of the target object was deemed to be an important instructional attribute for 73.71% of the instructions. Similarly, the specification of spatial relations was considered important in 66.25% of the instructions, while the pointing gesture was considered as important in 67.14% of the instructions. Similar to word-length, a higher proportion of correct human responses corresponded with a higher average rating for the importance of target color and spatial references. However, the average rating for the importance of pointing gestures revealed a separate and distinct trend. The average importance score for pointing was fairly uniform across instructions with 1, 2 and 3 correct responses, while it was lower for instructions with no correct responses.

Overall, our empirical findings demonstrate that the COSM2IC dataset provides a greater range of instructional ambiguity and employs a wider range of input modalities across its curated scenes.

In addition, the elicited human *instructee* responses suggest that certain attributes, of both the scene and the verbal instruction, are useful indicators of the corresponding task complexity—these findings help drive the design of the TCOP complexity predictor module (Section 5.6.1).

5.2.3 Baseline 1 - Human Performance

Study 2 establishes a competitive baseline in terms of human comprehension capability. On the block-world-based instructions (CB and UB) in the composite COSM2IC dataset, human subjects obtained an average comprehension accuracy of 83.53%, across both cluttered (82.42%) and uncluttered (84.89%) arrangements. For instructions involving real-world objects (CR and UR), human subjects obtained lower comprehension accuracy of 73.39 %, across both cluttered

(70.17%) and uncluttered (77.81%) arrangements.

5.3 Multi-modal comprehension models

As a prerequisite of the COSM2IC optimization paradigm, we first describe the creation of multiple DNN-based models, embodying different points on the accuracy-vs.-complexity curve, for comprehension of target acquisition instructions. We note that the currently available models are NOT designed for pervasive device-based deployment and execution, and thus, systematically describe the modifications made (across all models) to support on-device execution. Table 5.3 enumerates the different modality-specific characteristics and summarizes micro-benchmark performance results observed for each model, across both (cluttered, uncluttered) block-world setups.

5.3.1 Baseline 2 - RealGIN: A Non-Pervasive Model

We use the RealGIN model [129] as the representative, state-of-the-art model for real-time, multi-modal table-top instruction comprehension. However, we noted that this model utilizes *only* verbal and visual cues. Therefore, we first extended RealGIN to accept depth image-based inputs as well. RealGIN employs a RESNET [43] based backbone for extracting features from the image, a bi-directional LSTM for extracting language features from the verbal instruction, an adaptive feature selection module that identifies image features that are relevant to the text instruction, and a multi-modal attention mechanism (GARAN) to facilitate language-guided visual attention. On our COSM2IC dataset, we obtain the following performance for RealGIN:

- *Accuracy:* An average comprehension accuracy of 81.66% (s.d.=36.11%); with the accuracy on CB (78.87%) being lower than that for UB (84.46%).
- *Latency:* RealGIN's use of a RESNET-based backbone makes it incompatible for execution on resource-constrained pervasive devices. In particular, the RealGIN model cannot

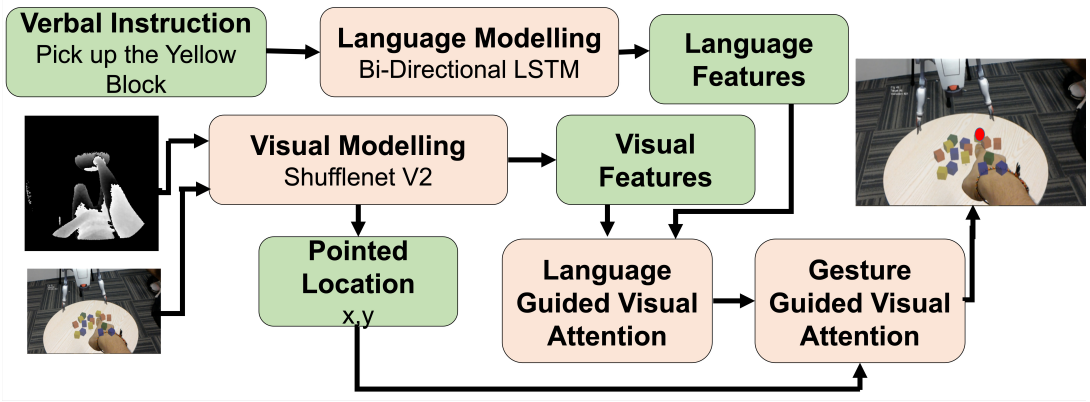


Figure 5.7: RealG(2)IN-Lite architecture

be loaded into the Jetson Nano device due to lack of sufficient memory; on a higher-resourced NVIDIA Jetson TX2 device, it incurs an average latency of 330 msec (s.d.=30 msec).

5.3.2 RealG(2)IN-Lite: On-Device DNN Model

Table 5.3: Summary of Features and Performance of the Light-Weight Models on the UB and CB Parts of COSM2IC Dataset

Name	Visual pipeline	Verbal dimension	Gesture based ROI	Attention/ GARAN	#FPN	Accuracy %	Latency msec	Energy mJ	Memory GB
RealGIN	RESNET	Full	Yes	Yes	6	81.7	330(±30)	2310.0	4.83
RealG(2)IN-Lite	shufflenet	Full	Yes	Yes	6	78.8	155(±10)	852.50	0.68
RealG(2)IN-Verbalite	shufflenet	Reduced	Yes	Yes	2	71.2	135(±10)	742.5	0.60
RealG(2)IN-Visionlite	3-layer CNN	Full	Yes	Yes	2	55.2	58(±5)	209.0	0.45
RealG(2)IN-Superlite	3-layer CNN	Reduced	Yes	Yes	2	52.2	55(±5)	174.0	0.39
RealG(2)IN-Ultralite	3-layer CNN	Reduced	No	No	1	26.5	35(±4)	122.5	0.35

To overcome the problems with RealGIN, we developed a new RealG(2)IN-Lite model, as shown in Fig 5.7 that supports on-device execution on pervasive platforms with two key novelties: (i) utilizes 3 distinct modalities; verbal instruction, scene image, and pointing gestures, (ii) replaces the RESNET-based architecture for visual processing with Shufflenet V2 [71]. The Shufflenet V2 converts the combination of an RGB and depth image into a visual feature embedding F_v. Concurrently, a bi-directional LSTM converts the verbal instruction into a language embedding feature vector f_t. f_t is then used in combination with F_v by a *Language-guided Visual Attention* module, that outputs a spatial *2-D heatmap* denoting the likelihood of the referred object's location. A new 2-layer multi-layer perceptron takes F_v as an input to pre-

dict the user's pointed location, which is fed to a final *Gesture-guided Visual Attention* module. The final module combines the visual heatmap and pointing gesture's RoI to generate the final coordinates of the selected object.

Model Training: To train the RealG(2)IN-Lite, we used 70-30 train-test split from UB and CB parts of the data. A pre-trained model on ReferIT dataset is used and then fine-tuned on the COSM2IC dataset using an Adam optimizer.

5.3.3 Lower Complexity Variants of RealG(2)IN-lite

RealGIN [129] comprises a deep CNN network for visual backbone, bi-directional LSTM for language features, 3 attentional modules, and 6 feature pyramid networks. We make 4 RealG(2)IN-lite models of low complexity by reducing these computational modules.

1. **RealG(2)IN-Verbalite**: Verbal pipeline is diminished by reducing the dimension of the embedding space for the LSTM from 128 to 64.
2. **RealG(2)IN-Visionlite**: Only the visual pipeline is modified to use a custom 3-layer CNN instead of the Shufflenet model of RealG(2)IN-Lite.
3. **RealG(2)IN-Superlite**: Computational complexity along BOTH the RealG(2)In-Visionlite and RealG(2)IN-verbalite pipelines is reduced.
4. **RealG(2)IN-Ultralite**: Attention mechanism is dropped and uses only 1 Feature Pyramid Network (FPN) module. Hence, it does not possess the capability to use the gestural input to define a RoI around the pointed location.

5.3.4 Model Performance:

The performance of RealG(2)IN-Lite and its lighter variants is summarized in Table 5.3. For each model, the accuracy is measured over the block-world based instructions. Latency and

energy were measured on Jetson Nano using *jetson_stats* library. Latency is defined as the average execution time over 1000 randomly selected instructions. Memory is measured by loading individual models, using the PyTorch API, on a Dell PowerEdge R740 server.

Table 5.3 shows the inherent accuracy-vs.-latency and accuracy-vs.-energy trade-offs across the models. RealG(2)IN-Lite achieves $\sim 50\%$ reduction in inference latency, consuming 37% energy and 15% of memory as compared to RealGIN, while losing out $\sim 3\%$ accuracy. The variants achieve lower accuracy but with a significant reduction in latency, energy, and memory overhead–e.g., compared to RealG(2)IN-Lite, RealG(2)In-superlite achieves 50+% accuracy with a 5-fold reduction in energy and a 6-fold reduction in latency.

The accuracy-vs.-overhead also reveals that the final COSM2IC system does not need to consider all variants. For an example, between RealG(2)IN-Lite and RealG(2)In-verbalite, there is not much change in latency/energy but there is a considerable 7% drop in accuracy. Also, RealG(2)In-superlite achieves similar accuracy as RealG(2)In-visionlite along with a small reduction in energy and latency. Thus, we omit RealG(2)In-verbalite and RealG(2)In-visionlite from further analysis.

Overall, the results demonstrate that the simplification of pipelines along different modes result in different trade-offs. If we *predict* the task instances where either RealG(2)In-superlite or RealG(2)In-ultralite perform correctly, it can significantly reduce the energy and latency overheads. Next, we describe how the COSM2IC system implements this principle in practice.

5.4 COSM2IC: Realizing dynamic model switching

COSM2IC is based on the observation that a significant proportion of instructions may be accurately comprehended by less complex models, and the inference complexity/latency may be considerably reduced if such instructions could be selectively processed by lower-complexity variants. Figure 3.1 shows the architecture of the implemented inferencing pipeline. For any given multi-modal 'target acquisition' instruction, COSM2IC uses one of the models (RealG(2)IN-

Lite or RealG(2)IN-superlite or RealG(2)IN-ultralite) depending on the instructional and environmental context.

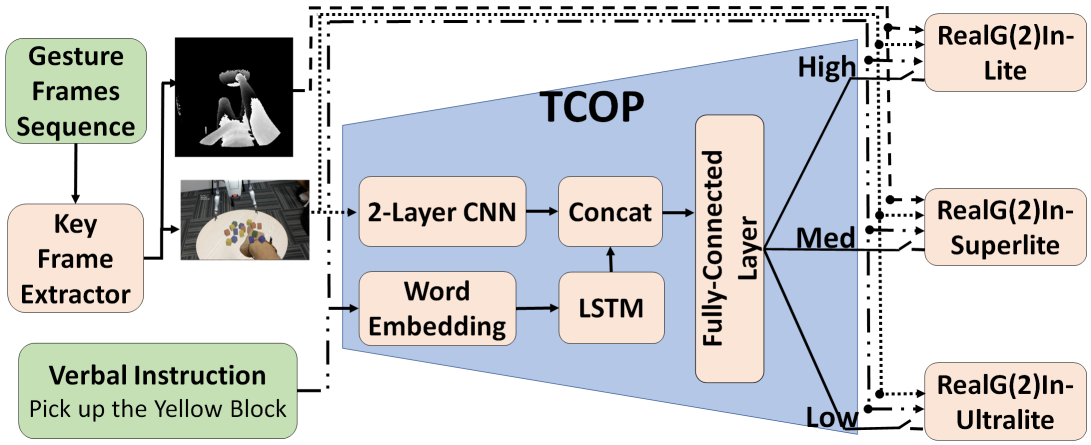


Figure 5.8: Architecture of the proposed COSM2IC system

The COSM2IC supports *continuous* capture and comprehension of human-generated instructions. When a multi-modal instruction is expressed, it employs a key-frame extractor to identify the most useful frame (one that best captures the objects and the pointing gesture). This key-frame is then used along with the verbal instruction to ascertain the task complexity, and then fed as input to the appropriate model to infer the target object. The lightweight key frame extractor imposes a negligible latency of $\leq$ 1msec and sustains processing of depth sensor frames at speeds up to 1000 FPS (exceeding the HoloLens sampling capability), while consuming only 300mJ energy.

5.4.1 Task Complexity Predictor (TCOP)

The Task Complexity Predictor (**TCOP**) serves as a demultiplexer, taking the multiple instructional inputs and redirecting them to one of three comprehension pipelines. To work effectively, it has to satisfy two key properties: (a) *Lightweight Execution:* introduce only minimal overhead and delay in the comprehension process, and (b) *High Precision:* should invoke one of the shallower processing pipelines only when it is very confident that it can accurately perform the comprehension. High *recall*, while desirable, is not critical: in case, TCOP incorrectly judges the task complexity to be high, its invocation of a more complex model will cause unduly high latency but not degrade the comprehension accuracy.

To support these goals, we developed a computationally inexpensive *Neural network-based* approach that uses visual and language-related features found in the instruction to predict task complexity. TCOP uses a lightweight LSTM of hidden embedding size of 32 for the language features and the visual features are encoded using a 2-layer simple CNN network. It combines the two modalities using a single fully-connected layer to classify the 3-way complexity. Only the RGB image is used for complexity estimation and does not use the depthmap. For each instruction, the desired output label (task complexity) was annotated by observing the individual outputs of RealGIN variants. For a given instruction, if it was possible to obtain the correct output via RealG(2)IN-ultralite, then the task complexity is annotated to be *'low'*. If the correct output is obtained from RealG(2)IN-superlite and not from RealG(2)IN-ultralite, then the task complexity is considered to be *'medium'*. If neither RealG(2)IN-ultralite nor RealG(2)IN-superlite could provide correct output, desired output label is said to be *'high'*. We corroborated this objective annotation of ground-truth of complexity with human-annotated subjective difficulty scores from study 2. Since the subjective scores were on a scale of 0-4 of *Easiness of Task* we took scales 0-1 as high complexity, 2 as medium complexity and 3-4 as low complexity. We observed a Jaccard similarity of 0.78 between the subjective and objective scores. With low execution latency (5msec) and low energy overhead (5 mJ), TCOP serves as a lightweight and efficient preprocessor for COSM2IC.

5.5 Prototype System Implementation

We have implemented a prototype version of a COSM2IC-based Comprehension Engine, and deployed it on multiple pervasive devices, including:

- *NVIDIA Jetson Nano [10]*: The Jetson Nano serves as an exemplar of a low-cost, embedded platform (with built-in support for executing DNN models) that can be placed within a future pervasive device, such as a mobile robot or kiosk. The Nano device comprises a 128-core NVIDIA GPU, a Quad-core ARM processor and system memory of 4GB, and supports a custom Linux-based OS called *Linux4Tegra* that is flashed onto a 64GB SD

card. While the Nano cannot execute the RealGIN model (too heavyweight), it is able to execute all our proposed models.

- *HoloLens-based Comprehension:* We also attempted to deploy COSM2IC on the HoloLens via the Windows Machine Learning (WinML) API and ONNX model format. However, WinML currently does not support some of the required multi-modal functionality (e.g. multi-modal inputs, language guided attention). As a workaround, our HoloLens prototype currently offloads sensor data, in real time, to a COSM2IC implementation on a Jetson Nano.

- *NVIDIA Jetson TX2 [10]*: To provide a more comprehensive evaluation, we also implemented the Comprehension Engine using the NVIDIA TX2, which possesses a higher-end 256-core GPU,a dual-core NVIDIA processor and 8 GB system memory and is capable of also executing the Resnet-based default RealGIN model.

5.6 Evaluation

We now present empirical performance results of COSM2IC and its constituent components, thereby illustrating the superiority of our proposed approach for real-time, low-power and accurate instruction comprehension. We studied the performance of COSM2IC on the block-world and realistic-object parts of the dataset separately. Accordingly, we trained the models separately on the block-world data (CB and UB) in the COSM2IC dataset collected in study 1. We used 70% (1796) of the instruction corpus as the training split, with 30% (770) used for testing, after ensuring that an equal mix of cluttered and uncluttered setups are included in the data. Similarly, for studying the performance on the CR and UR parts of the dataset with more realistic objects, we used 357 instructions for training and 153 instructions for testing. To measure accuracy, we used the mid-point of the bounding box estimated by the various comprehension models; task comprehension is deemed to be accurate if this mid-point lies within the bounding box of the true target object. We chose to use this metric since the COSM2IC dataset is focussed towards enabling table-top acquisition task where we predict a single x,y

Table 5.4: Model Performance on the (CB + UB) Instructions

Model	Full Dataset	Complexity High	Med.	Low	Latency(ms)		Energy(mJ)	
	Accuracy(%)				Nano	TX2	Nano	TX2
RealGIN	81.7	77.8	78.4	85.1	N.A	330(30)	N.A	2310
RealG(2)IN-Lite	78.9	77.0	77.7	82.2	175(12)	155(10)	787.5	852.5
RealG(2)IN-superLite	52.2	55.8	55.9	59.1	72(6)	51(5)	144	174
RealG(2)IN-ultraLite	26.5	24.8	26.7	28.0	55(6)	35(4)	137.5	122.5
COSM2IC	76.1	73.7	77.8	78.5	134(17)	110(15)	554	590

target location. Unless otherwise stated, system performance metrics, such as power and latency, are obtained as averages over 1000 distinct runs, and are evaluated using the Nano, our representative pervasive computing platform.

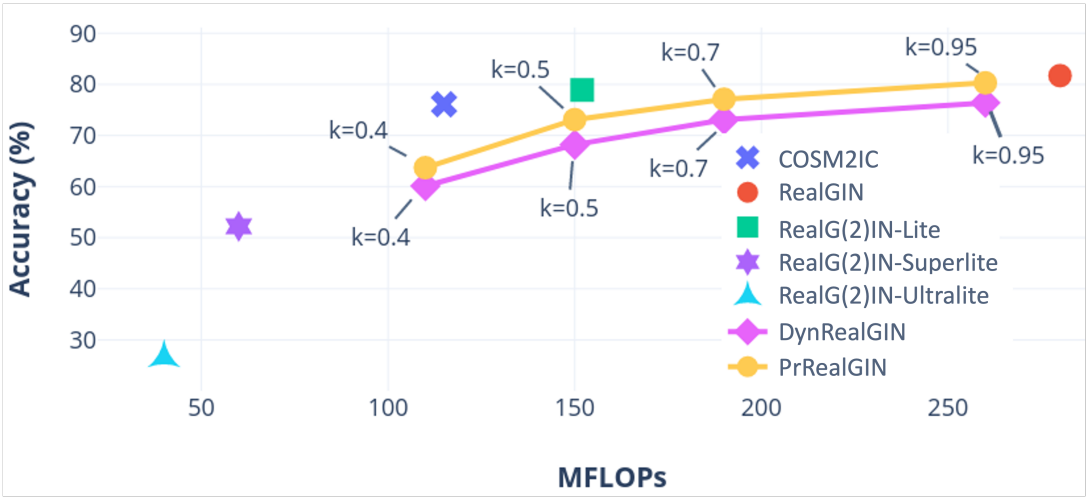


Figure 5.9: Computations vs Accuracy comparison of various comprehension models

5.6.1 Task Complexity and TCOP Performance

The COSM2IC testing dataset contains 770 multi-modal instructions, corresponding to 40 distinct configurations, of which 201 (26.10%) are marked as *Low-Complexity* tasks (i.e., those where executing RealG(2)IN-ultralite provides accurate comprehension), 148 (19.22%) are marked as *Medium-Complexity* tasks (i.e., those where executing RealG(2)IN-superlite provides accurate comprehension while RealG(2)IN-ultralite goes wrong) and the remaining 421 are marked as *High-Complexity*. We used this classification as ground-truth to determine whether TCOP is able to precisely switch between the models to save energy and latency with

minimal effect on accuracy. Overall, we observed that TCOP accomplishes our goal of high precision selection of lighter models. TCOP predicted that 275 of the 770 instructions could be processed by either RealG(2)IN-superlite or RealG(2)IN-ultralite. Among these 275 instructions, the ground-truth complexity of 228 were indeed marked as low or medium. Specifically, we found that TCOP's precision when switching to RealG(2)IN-superlite is 78.65%, and its precision in switching to RealG(2)IN-ultralite is 85.11%. Among the 421 high-complexity instructions, all three models performed the target inference inaccurately for 106 instructions.

5.6.2 COSM2IC's Performance on Instruction Comprehension

Table 5.4 summarizes the accuracy (on the CB and UB data subsets), latency and energy overheads of COSM2IC against our baselines. In the column for the latency, value specified in the brackets refers to the standard deviation. We make the following key observations:

- All the models achieve higher accuracy for lower complexity tasks. In comparison to RealG(2)IN-Lite, COSM2IC suffers only a modest ($\sim$2.7%) reduction in accuracy, but is able to achieve a substantial $\sim$23.4% and $\sim$21.2% reduction in processing latency on the Nano and TX2 devices respectively. COSM2IC also achieves energy savings of $\sim$22% on the Nano and TX2 devices respectively. The baseline RealGIN model cannot be executed on the Nano device, due to inadequate memory capacity.
- While RealGIN does load and execute on the resource-richer TX2 platform, its computational latency, energy and memory overheads, are 3x, 4x and 3x higher respectively, than our proposed COSM2IC approach. In spite of these significant savings, the accuracy of COSM2IC is only $\sim$5% lower than the RealGIN model.

Collectively, the results not only demonstrate the superiority of the adaptive COSM2IC approach, but also illustrate why non-adaptive approaches are unable to simultaneously achieve both low latency and high accuracy.

We also evaluated whether the trends observed above for the UB and CB data subsets would hold in setups involving virtual kitchen objects used in CR and UR subsets of the COSM2IC dataset. For this evaluation, we retrained all the models on the UR and CR datasets using the same procedure. It must be noted however that there are only about 500 instructions in the CR and UR datasets and hence the training and testing datasets are about 5 times smaller than the block-world datasets. Here again, we observed that the accuracy of COSM2IC is only marginally ($\sim$2%) lower than RealGIN and almost the same as the accuracy of RealG(2)IN-Lite. While the latency and energy consumption of COSM2IC is slightly higher than what was observed in Table 5.4 for the block-world based instructions, COSM2IC continued to achieve about nearly 3-fold reduction in latency and energy consumption in comparison to RealGIN. These results show that the COSM2IC approach may indeed be useful in more generalized settings such as the virtual kitchen, involving higher diversity of objects. However, it must be noted that CR and UR datasets do not completely depict a real-world object scenario since the target objects are virtual. In future, as an extension to our existing datasets we intend to include setups that include diverse real-world objects.

In general, the COSM2IC paradigm could be easily extended to accommodate more than just the 3 DNN based models discussed above, including those based on conventional machine learning approaches such as [109]. The gains in latency and energy achieved by COSM2IC is despite the fact that nearly half of the instructions in our corpus fall under the high-complexity category. In more simpler,clutter-free object arrangements as studied in [109], one may expect a higher proportion of low-complexity instructions, which could imply more gains in latency and energy as COSM2IC switches to lighter models (including the Bayesian approach in [109]) more often.

5.6.3 Superiority over Alternate Model Optimization

To compare against alternate DNN optimization approaches we chose (i) the classical static pruning approach introduced in [13] to identify and eliminate redundant nodes according to

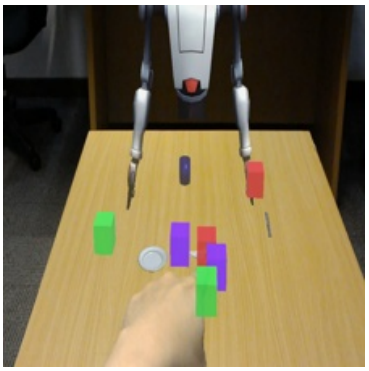


Figure 5.10: Original Image

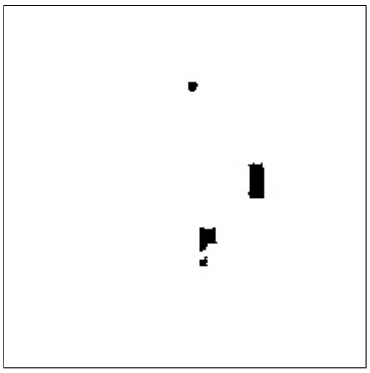


Figure 5.11: Sparsity $k = 0.8$

Figure 5.12: Sparsity $k = 0.6$

Figure 5.13: DynRealGIN sparse convolution mask for different sparsity values

a specified a sparsity parameter ($k \in [0,1]$ specifies the fraction of nodes to preserve) and (ii) the dynamic DNN optimization introduced in [101] where a set of *masks* that correspond to portions of a given input image are determined at inference time, so that convolutional filters can be applied only over these specific portions of the image (k specifies the fraction of pixels included in convolution operations).

In this work, the statically pruned model, PrRealGIN, is obtained by removing redundant nodes from RealGIN, while the dynamically optimised model is called DynRealGIN. Figure 5.9 shows their complexity-vs.-accuracy performance evaluated on UB and CB datasets. Since the pruning approach is currently unsupported on the Jetson platform's PyTorch implementation, we compare their overheads in terms of total computational complexity (FLOPs). The degradation in performance of DynRealGIN and PrRealGIN is found to be more severe at higher levels of sparsity (lower k). Most importantly, COSM2IC *outperforms either of these conventional model optimization techniques*: under roughly comparable computational complexity ($\sim$106-110 MFLOPs), COSM2IC's comprehension accuracy (76.13%) is $\sim$16% higher than for DynRealGIN and $\sim$12.4% higher than PrRealGIN.

Figure 5.13 helps us visualize the mask of DynRealGIN pipeline for varying sparsity values for a representative image. The black areas in this figure show the masked off parts where DyNRealGIN skips convolution operations, thereby reducing latency and energy. However, reduced sparsity increases the likelihood (especially for cluttered scenes) that areas of high relevance (either locations where the target object is found, or which are referred to by spatial

relations in the verbal instructions) are erroneously excluded from convolution operations early in the execution pipeline.

5.7 Discussion

While SoftSkip has been primarily applied to language+vision-based REC tasks, and COSM2IC is designed for language+vision+gesture-based REC tasks, we observe a significant synergy between the two approaches. Both approaches introduce dynamic model optimizations, where a cost-effective computation (skip values in SoftSkip and TCOP in COSM2IC) determines the execution pathway for the grounding task. A notable distinction between the two lies in the determination of the execution pathway: SoftSkip solely utilizes language features for computing skip values, whereas COSM2IC evaluates task complexity using both language and visual features.

We further anticipate that the principles outlined in COSM2IC remain applicable even with advancements in REC models. We contend that the three models employed to address high, medium, and low complexity tasks can be readily substituted with more recent and accurate models as they become available.

5.7.1 Achieved Design Goals

Following the work of SoftSkip, which addressed the low-latency and battery-powered sense-making design goals, the challenge of achieving multi-modal sense-making remained. To tackle all three design goals simultaneously, we introduced COSM2IC. For multi-modal sense-making, COSM2IC employs a combination of language and pointing gestures to capture human instructions. The pointing gestures are captured using a depth map from the LiDAR sensor in the smart glasses.

The COSM2IC approach, along with other baselines, maintains comprehension latency well

below the intended 1 second for low-latency sense-making. Additionally, the proposed dynamic model optimization through COSM2IC reduces processing energy consumption by nearly 30% while ensuring that the degradation in task accuracy, compared to the baseline on-device pervasive model RealG(2)In-Lite, does not exceed $\sim$5%.

The chapter focused solely on processing energy overhead and did not consider the potential increase in sensing energy overhead that additional sensors like LiDARs might introduce. However, in the upcoming chapter, we plan to expand our dynamic model optimizations to address reductions in sensing energy overhead as well.

5.8 Summary

In summary we explored the COSM2IC approach, which demonstrate a dynamic model optimization to reduce the **average** *inference time*|| *complexity* ||*energy* in multi-modal REC with vision, language and pointing gesture-based REC. COSM2IC uses an alternative *switching* based resource optimization paradigm, where the inference process dynamically switches (on a per-instance basis) between the execution of multiple available models, each catering to a different level of complexity *along different modalities*. The approach is motivated by our observation that individual task instances, in the real world, have widely varying complexity trade-offs across different modalities, depending on a combination of *environmental* and *instructor* context. Key to COSM2IC's viability is the development of a cheap and lightweight TCOP pre-processing module that can; (a) rapidly estimate the instructional and environmental complexity for each mode, and (b) determine the *right model* (least complex one which achieves accurate comprehension) to be executed. Therefore, this work makes the following **contributions**,

- *Multi-modal ego-centric instruction dataset:* We have curated a unique and large dataset of multi-modal referring instructions for the table-top scenario from a first-person viewpoint, using a head-mounted camera. The dataset is based on a careful placement of

table-top objects such that they induce different levels of ambiguity and multi-modal variations. Notably, we also capture a depth image along with other inputs.

- *Improved and diverse multi-modal comprehension models:* To support COSM2IC's dynamic model-switching paradigm, we build a set of multi-modal (verbal, visual, gestural) models with different complexity levels. We enhanced the state-of-the-art RealGIN model by; (a) replacing its visual backbone with a ShuffleNet-based pipeline and (b) extending attention modules to accept depth-image-based gesture input. The enhanced RealG(2)IN-Lite model offers 2x improvement in processing latency and uses 7x less memory for a meager 3% loss in accuracy. To support comprehension in even lower-resource situations, we further develop two lighter variants of RealGIN-based models.

- COSM2IC *paradigm for efficient multi-modal REC:* We introduce a new model switching paradigm to reduce the instruction processing latency 3-fold (compared to RealGIN while achieving a target object identification accuracy of 76.13% (only 5.53% lower than RealGIN). COSM2IC achieves this with the help of a lightweight, low-latency neural network-based model for TCOP that uses a combination of visual and language embedding features to classify different task complexity levels, which chooses a specific multi-modal inference model for execution. Further, COSM2IC outperforms prior complexity-reduction approaches, offering $\sim$12-16% higher comprehension accuracy under equivalent computational complexity.

Chapter 6

Jointly Reducing Processing and Sensing Energy Overheads with CAS Paradigm

With SoftSkip and COSM2IC, we have showcased the means to reduce processing energy and latency by utilizing dynamic execution pipelines. However, processing energy still accounts for a part of the overall energy footprint. For instance, COSM2IC uses a depth camera which accounts for an average power consumption of 2.5W as sensing energy footprint, which is 33% of the overall energy footprint of executing multi-modal human instruction comprehension model on a Jetson TX2 device. In this study, we investigate a dynamic optimization paradigm called the *Commit-and-Switch* (CAS) paradigm. This approach leverages the depth camera as an alternative sensor for capturing visual context, activating it only when necessary. By employing this dynamic model optimization and utilizing the depth camera for visual context, we effectively reduce both sensing and processing energy overheads while minimizing the impact on overall accuracy.

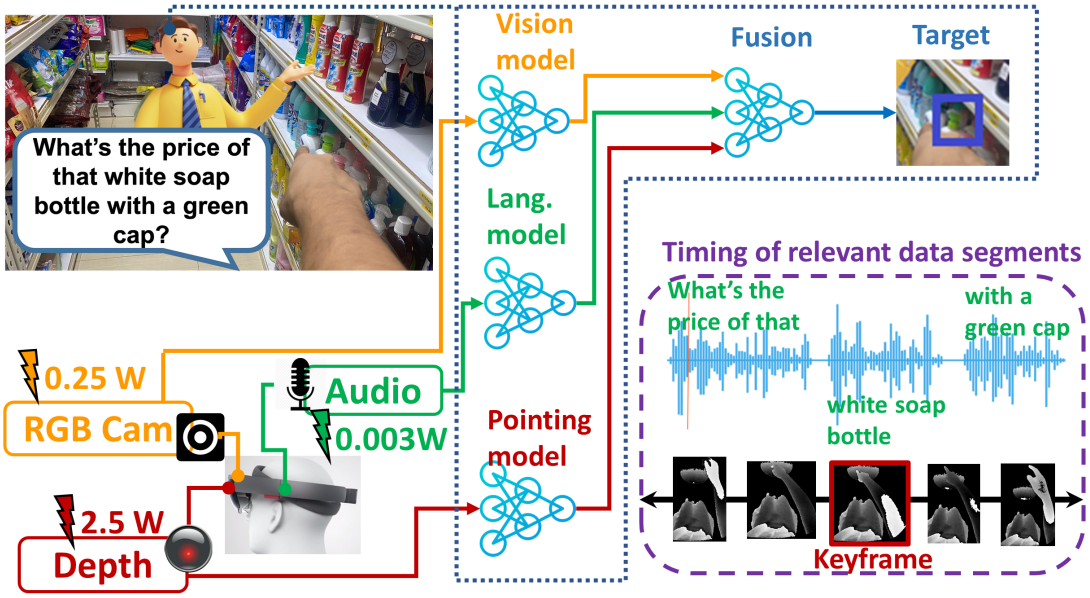
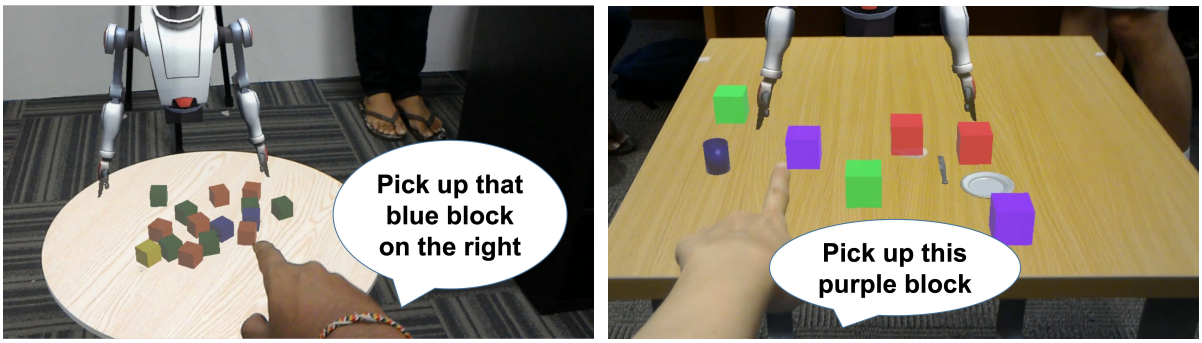


Figure 6.1: Motivating application- A virtual shopping assistant

6.1 Introduction

In this chapter, we again focus on the problem of energy-efficient "on-device" execution of multi-modal human instruction comprehension for object acquisition tasks. In particular, we aim to tackle two key challenges, (a) minimising the computational footprint of the multi-modal inference pipeline (which also reduces the inference latency) to fit it into a resource-constrained pervasive device, and (b) simultaneously reducing the energy-cost by minimizing the avoidable use of energy-hungry sensors (especially LIDAR sensors). To tackle the former challenge, we have explored COSM2IC in chapter 5. While COSM2IC significantly reduce the overall processing energy and latency, it omits the effect of sensing energy from the overall energy footprint. Thus, we will now delve into an alternate dynamic optimization technique to jointy optimize for both sensing and processing energy with minimal effect on the accuracy.

For example, in an uncluttered environment (Figure 6.2(b)), a rough estimation of pointing direction together with RGB scene analysis may be sufficient for identifying the target-object, whereas a more cluttered environment (Figure 6.2(a)) may require RGB scene analysis plus more complex parsing of the longer verbal command as well as precise, depth sensor-based, estimation of pointing coordinates. However, existing approaches explored in 5 along with COSM2IC do not actually optimize the *sensing energy* overhead. More specifically, consider



(a) Cluttered setup (b) Uncluttered setup

Figure 6.2: Diversity in multi-modal instructions

the figure 6.1. In this, the depth sensor needed for accurate pointing resolution is very energy hungry, consuming ∼10x the power of on-board RGB and microphone sensors.

We thus, propose a new paradigm, called *Commit-and-Switch* (**CAS**), designed to simultaneously reduce both *sensing* and *inference* overheads associated with the on-device execution of such multi-modal DNNs. The core CAS concept involves the use of triggered sensor activation, whereby more energy-hungry sensors are activated on demand, only if deemed necessary, *based on the complexity of the current task instance*. Past approaches for selective/dynamic triggering of sensors–for example, for energy-efficient gesture detection [80] or automated food journaling [88]–trigger a more expensive sensor whenever a cheaper sensor's value satisfies certain predicates. This 'simple predicate' approach is *inadequate* for tasks such as multi-modal instruction comprehension for two important reasons:

1. Determining the task complexity, and thus the need for specific sensors, itself is non-trivial and may require different sensing modalities as well as its own separate black-box DNN (e.g., see [107]). For example, the complexity of a multi-modal instruction may arise either due to visual factors, such as a cluttered environment, or linguistic factors, such as long instructions containing implicit references to multiple objects.

2. The "interval of relevance" for each individual sensor may vary dynamically, across different task instances. For example, the pointing gesture duration can not only vary significantly (fleeting to more durable), but its (start, end) points may also occur either before, after, or straddle the corresponding verbal command. As illustrated in Figure 6.1, the 'keyframe' of the depth sensor (i.e., the precise time instant where the user's hand/fingers

point at the target) occurs *before* the verbal command has been completed. Accordingly, using verbal instruction complexity to trigger the depth sensor may miss the pointing gesture, even though the microphone is the most energy-efficient of the three sensors in Figure 6.1.

To overcome these challenges and derive the benefits of triggered sensing, CAS unifies the process of complexity determination and task inference into a single "graybox" DNN pipeline with multiple complexity-driven processing branches (associated with dynamic sensor triggering) and heads, identified via a *principled* cost-benefit analysis technique. As an exemplar for CAS, we develop RealGIN-MH, an end-to-end CAS-based model for target acquisition instruction comprehension, and use a benchmark COSM2IC dataset ([107]) to demonstrate CAS's superior performance across task instances of varying complexity.

6.2 CAS: Overview

We now briefly touch upon the traditional sensor triggering approach (Figure 6.3) and the dynamic DNN optimization approach (Figure 6.4) and contrast them with our proposed Commit-and-Switch (CAS) paradigm (Figure 6.5). In Figures 6.3, 6.4 and 6.5 the sensors are denoted by S1, S2 etc. and their corresponding font colors represent their power consumption (Green $\rightarrow$ Low power, Orange $\rightarrow$ Medium power and Red $\rightarrow$ High power). Similar color codes are used for branches representing data processing.

6.2.1 Sensor triggering

In the traditional triggered sensing approaches (Figure 6.3), the sensing context is determined efficiently (i.e., with low energy overhead) using a low-energy sensor (S3), coupled with an energy-efficient context detector. In many cases, this context is used either to abort the processing pipeline (e.g., Jigsaw [68], e-Gesture [80]) or to select a specific downstream processing

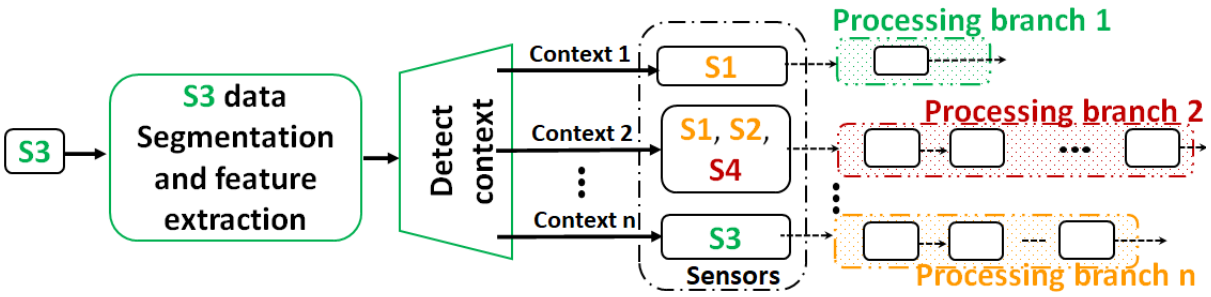


Figure 6.3: Typical sensor triggering for pervasive applications

branch, each of which imposes its own additional sensing and inferencing overheads. Once a certain branch is selected, the corresponding streams of sensor data are then synchronized to perform the inferencing on subsequent data frames. Clearly, the inability to perform such synchronization (e.g., if some of the task-critical sensor data from S4 are missed due to a delay in triggering the sensor) will adversely affect the inference accuracy.

6.2.2 Dynamic model optimization

The dynamic model optimization paradigm, shown in Figure 6.4, assumes that the sensors (S1 - S4) are always on and thus focuses solely on minimizing the *inferencing* overhead. In this paradigm, the initial context detection is used to select one of multiple candidate processing branches, each differing in the complexity of computations performed by the processing blocks. We note that the initial context detection itself may not be extremely lightweight (and is thus indicated in yellow), as it may involve the execution of DNN models. The COSM2IC system [107] discussed earlier exemplifies this paradigm. This approach is especially suitable for perception tasks executed on resource-constrained, but *powered*, IoT/edge devices (e.g., object detection being performed by a wall-mounted, micro-controller driven CCTV camera system), where the processing bottleneck is more critical than the sensing energy overhead.

6.2.3 Commit-and-Switch (CAS) paradigm

Our proposed CAS paradigm, illustrated in Figure 6.5 aims to combine the benefits of both the above approaches for multi-modal, multi-DNN inference. Unlike the paradigm of Figure 6.3,

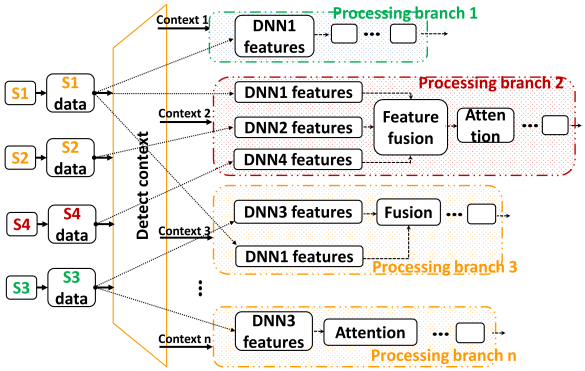


Figure 6.4: Dynamic DNN optimization (COSM2IC)

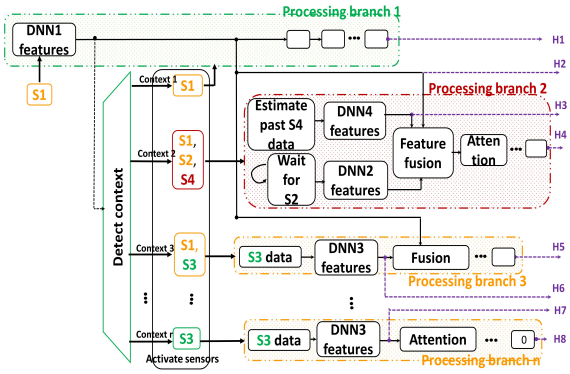


Figure 6.5: Commit-and-Switch (CAS) paradigm.

the determination of the sensing context (task complexity for our exemplar application) is not a distinct step, but is *integrated* into the DNN-based inference pipeline. This context detector can conceptually utilize an intermediate state from any layer of the DNN processing pipeline, chosen so as to exploit problem-dependent trade-offs between efficiency and accuracy. In CAS, we first *commit* to a processing branch (branch 1 in the figure) that depends on a low/medium power sensor(s). The energy-intensive sensors and their corresponding processing pipelines are inactive at this point. Even as the processing in the initial committed branch goes on, CAS piggy-backs on the DNN features already generated in this branch to make a classification of the task context; consequently, context determination is considered to be relatively low energy (Green). This context is then used to potentially switch to other processing branches (e.g., branch 2), which may require the activation of corresponding additional energy-intensive sensors; else, the initially committed branch is executed in its entirety without activating any additional sensors. Like the on-demand triggering paradigm in Figure 6.3, CAS must also accommodate the likelihood that even a modest triggering latency can cause task-critical sensor data (say from S4) to be missing. We note, however, that in multi-modal sensing, the likely correlation across sensor observations raises the possibility of *estimating* the missing data of a sensor from the currently-available data stream of other sensor(s). This is reflected by the processing block "Estimate past S4 data" in branch 2. The reverse situation, illustrated by the block "Wait for S2" in the figure, whereby some sensor data may not be readily available is also possible. For example, if the user issues a long verbal command, the inferencing task may need to wait until the verbal instruction is complete.

6.3 Characteristics of Multi-modal instructions and setup

We use our proposed COSM2IC dataset as the representative setup for multi-model object acquisition task instructions. Details of the COSM2IC dataset is extensively studied in chapter 5.

Sensor Energy Profiles: To determine judicious choices for different inference branches, we also need to quantify the relative energy overheads of the different sensors. We used RealSense L515 [4] as our representative depth sensor in our evaluations. Measurements performed using a Monsoon power monitor revealed that RealSense consumes $\sim$2.5W of power for capturing depth frames, which was nearly 10x higher than the operating power of a typical RGB camera. As we shall later see (Section 6.5), this implies an energy consumption of about 388 mJ (nearly half of the inference energy of the RealG(2)In-Lite model) if the depth sensor is active for a duration equal to the average execution latency of the RealG(2)In-Lite model on the COSM2IC dataset, when evaluated on a Jetson TX2 device. An analysis of the dataset reveals the following characteristics that will influence the choice of different branches and triggering sensors:

- *Possible Sensing Redundancy:* The location indicated by the pointing gesture can be sensed either via the RGB camera or via the depth camera. While the accuracy of pointing resolution is higher when depth information is used [34], it is possible that, under conditions of low scene clutter, the target object may be distinguished by just using the less precise RGB camera-based pointing inputs. Hence, the depth camera, which has a significantly higher energy overhead, should be activated only on demand. Also, given this redundancy, it *may* be possible to regenerate past values of depth camera data using a combination of concurrent RGB data and depth frames acquired with a modest delay.

- *Optimizing Audio Sensing:* Broadly, the RGB camera sensor data is indispensable for the task, as any approach for target acquisition task requires RGB-based object detection. We note that it may be possible to determine the target object using just the RGB and pointing data *alone* (i.e., without the verbal input) in certain selected cases–e.g., when the

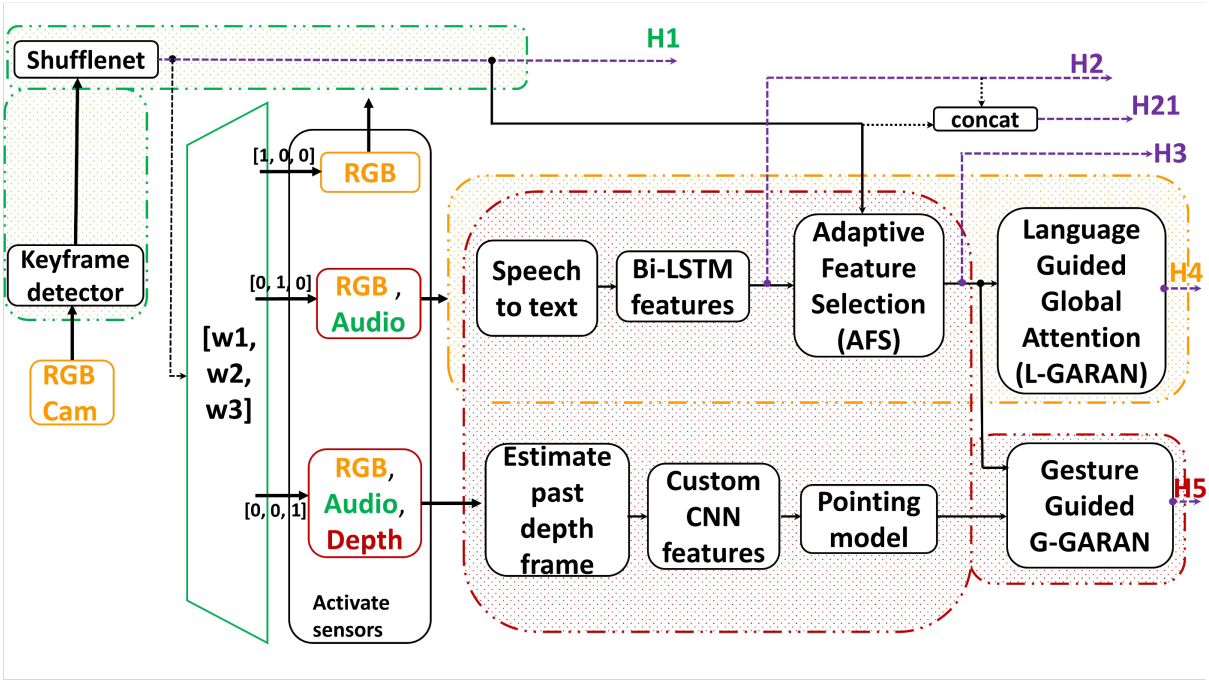


Figure 6.6: RealGIN-MH Architecture

scene has very low clutter (objects spaced far apart from one another), such that pointing itself unambiguously identifies the object. However, on-demand, delayed activation of the audio sensor is not feasible, as there is simply no alternate way to reconstruct past verbal instructions.

As the baseline models in our study we consider RealG(2)In-Lite (a statically pruned multi-modal human instruction comprehension model) and COSM2IC model (a dynamically optimzed model) proposed in chapter 5. In the case of COSM2IC, it does not support on-demand sensor triggering and assumes that all the sensors are active (even if they are eventually unused) throughout the inferencing process.

6.4 RealGIN-MH: CAS-based Inferencing

We now detail the design of RealGIN-MH, which employs the CAS paradigm to perform on-device multi-modal instruction comprehension. Figure 6.6 shows the exact sensors used in each of the three main processing branches involved in RealGIN-MH. The low complexity branch (enclosed by green dotted lines) performs the comprehension task using only the RGB

camera sensor data. This branch involves a key-frame detector that identifies a key-frame and a Shufflenet [124] visual backbone to extract features from the detected key-frame. Further processing blocks in this branch use these features to directly output the location of the target block (based solely on the pointed location from the key-frame) via output head **H1**. The features from the Shufflenet backbone are also used by the context detector (represented as the green demultiplexer box) to output a 3-element binary vector representing the branch chosen for subsequent execution.

If the context detector detects a low complexity context ([1,0,0]), the inferencing process continues along the low complexity branch. The medium complexity branch is activated for a context vector value of [0,1,0]) and is enclosed by the orange dotted lines in Figure 6.6. This block uses sensor data from both the RGB camera and audio sensor. The processing blocks involved in this branch are the speech-to-text module, the Bidirectional LSTM to extract features from the text, the Adaptive Feature Selection (AFS) module and the language-guided global attention L-GARAN (all explained shortly), which outputs the target object (head **H4**). Similarly, the high complexity branch is enclosed by the red-dotted lines in the figure. This branch uses all three sensors (RGB camera, audio and depth camera). This branch has significant overlap with the blocks in the medium processing pipeline as it also uses the text-to-speech module, Bi-LSTM and the AFS. Additionally, when processing the depth data, it first tries to *reconstruct* an estimate of the past depth-frame that is in sync with the RGB keyframe. The features from this depth image are provided to a pointing model. Finally, a gesture-guided G-GARAN module (instead of the L-GARAN used in the medium complexity pipeline) is used to output the target object location via head **H5**. Across all pipelines, the RGB and audio sensor (capturing verbal inputs) are always active (even though the low complexity branch does not utilize verbal cues), with RealGIN-MH focusing principally on dynamic activation of the energy-hungry depth sensor.

6.4.1 Keyframe extraction and Shufflenet backbone

Empirical observation shows that the most informative segment (which we call the "keyframe") in a pointing gesture corresponds to one where the hand is momentarily stationary, steadily indicating the target. Keyframe detection is done via a 4-layer CNN network with ReLU [77] activation for intermediate layers and Softmax activation for the final layer. This model, trained for 10 epochs using a balanced set of COSM2IC ground truth data, accepts an incoming RGB frame as an input and outputs its probability of being a key frame (class 0='not key', class 1='key'). During inferencing, an RGB frame is identified as a keyframe if class 1 probability is $\geq$0.8; the Shufflenet visual backbone then extracts the visual features used by subsequent processing blocks.

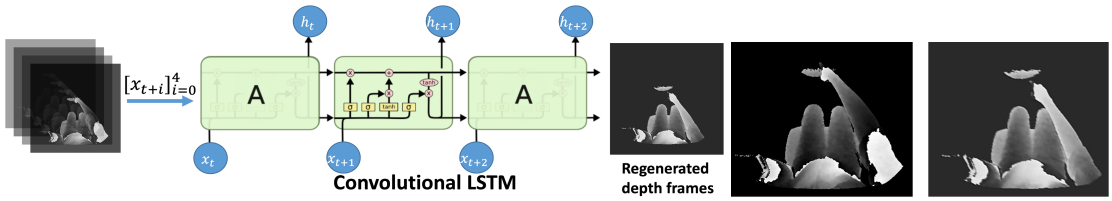
6.4.2 AFS and Language-guided Global Attentive Reasoning (L-GARAN)

We follow the same approach as explained in [129] for calculating AFS and L-GARAN features. The visual backbone computes features at different feature scales. Let us assume that these features are $F_{v1} \in \mathscr{R}^{m_1 * m_1 * s_1}$, $F_{v2} \in \mathscr{R}^{m_2 * m_2 * s_2}$, $F_{v3} \in \mathscr{R}^{m_3 * m_3 * s_3}$. $m_1 > m_2 > m_3$ refer to the resolutions of feature maps and s_1, s_2 and s_3 refers to the feature channels. Let language feature embedding computed with an LSTM be f_t. Then, AFS features are calculated as follows,

$$\begin{aligned} [\beta_1, \beta_2, \beta_3] &= F_{AFS}(f_t) \\ F_v &= \beta_1 * F_{v1} + \beta_2 * F_{v2} + \beta_3 * F_{v3} \end{aligned} \tag{6.1}$$

$\beta_1, \beta_2, \beta_3$ are determined from the f_t language embedding.

L-GARAN is a multi-modal attention component that uses language features as a pivot to compute a language attentional feature map F_{L-att} that identifies important regions in the visual feature map. This module is activated when $w_2 = 1$, and takes the AFS features as an input.



(a) DDPM architecture (b) Original image (c) DDPM output

Figure 6.7: Delayed Depth backProp. Model (DDPM)

6.4.3 DDPM: Delayed Depth Backpropagation

RealGIN-MH employs on-demand triggering for the energy-hungry depth sensor: once this module is activated, a trigger signal is sent to the depth camera to capture and stream $N = 5$ depth frames. After streaming, the depth sensor reverts to its low-power 'sleep' mode. The power overhead in 'sleep' and 'streaming' state is 1.5W and 2.5W, respectively. We experimentally observed an activation delay of ∼400 msec across 3 different commercial depth sensors (Leapmotion, Kinect DK and RealSense). This delay can cause the captured depth image frame to be significantly delayed from the *key* RGB frame, thereby resulting in incorrect pointing resolution. Therefore, we have developed the DDPM model to regenerate, using the 5 delayed depth frames captured on activation, the depth frame corresponding to the RGB keyframe. As shown in figure 6.7(a), we use a Convolutional-LSTM Encoder and Decoder model to perform this regeneration. Let D_t be the depth frame at the *key* frame, $D_{t'}$ be the first depth frame after the startup delay, $D_{(t'+1)}$ - $D_{(t'+4)}$ be the subsequent 4 depth frames and $D_t^{'}$ be the regenerated depth frame. Then, our encoder-decoder Conv-LSTM model is calculated as follows.

$$\begin{aligned} F_g &= Conv-LSTM_{Encoder}(\{D_{(t'+i)}\}_{i=0}^{4}) \\ D_t^{'} &= Conv-LSTM_{Decoder}(F_g) \end{aligned} \tag{6.2}$$

6.4.4 Gesture-guided Global Attentive Reasoning (G-GARAN)

The computation of this component is identical to L-GARAN except, instead of f_t language features, we use F_g (encoder output of DDPM) as a pivot. Intuitively, the model focuses greater

visual attention in the vicinity of the pointed location.

6.4.5 CAS - Selection of output heads

As previously discussed, the instruction comprehension pipeline consists of multiple modules that accept different modalities as inputs. The baseline RealG(2)IN-lite contains only one output-head (corresponding to H5 in Fig 6.6) that utilizes all of these sensor inputs and modules for every single input instruction. So as a first step in the CAS paradigm, we identify potential exit-paths that constitute the branches of processing pipelines, each offering varying accuracy and energy trade-offs. To determine the optimal branch points and compute heads, we propose an iterative training approach.

In this iterative training approach, we initially introduce $N = 6$ compute heads into the RealG(2)IN-Lite comprehension pipeline, as depicted in Figure 6.6. H1 – H5 represent various potential exit points from different processing blocks of the RealG(2)IN-Lite model, utilizing different sensor combinations. We also introduced a hybrid-branch H21, which concatenates the features for H2 and H1 and thereby uses both audio and RGB camera sensor data streams. Note that the context detector is disabled at this step. These compute heads are strategically selected to cover different endpoints of the comprehension pipeline, and each head is associated with an energy cost C_i. Here, C_i represents the sum of processing and sensing energy required for executing the i^{th} compute head. Subsequently, we iteratively train each compute head, following the forward computation order for each batch of data samples from the training set. This training process helps determine the IoU $A_i = IoU(pred, gt)$, where A_i quantifies the intersection over union (IoU) value between the predicted and ground truth bounding boxes. To select the optimal $K = 3$ heads from the initial set of N heads, we follow the following principles:

1. We always choose the head with the highest A_i to limit the drop in accuracy resulting from dynamic switching among different heads. Usually, this tends to be the head that involves the most energy-intensive and high-fidelity sensing and processing.

Table 6.1: Accuracy, cost and efficiency for various compute heads on COSM2IC dataset

	H1	H2	H21	H3	**H4**	**H5**
Cost	**0.3**	0.3	0.5	0.5	**0.5**	**10.9**
Accuracy	**0.67**	0.01	0.7	0.73	**0.74**	**0.78**
Efficiency	**2.23**	0.03	1.40	1.46	**1.48**	**0.07**

2. We compute the Efficiency, $E_i = \frac{A_i}{C_i}$. The remaining $K-1$ branches are then selected based on the highest efficiencies, achieving a balance between accuracy and energy cost. Usually, these are heads that can do the job far more efficiently than the most accurate head, for a significant proportion of the inputs, but fail when encountered with complicated inputs.

Table 6.1 provides the accuracy, cost and efficiency of each compute head. Heads that are marked in bold are the chosen K heads. Based on the CAS principle, we first chose H5 which yields the highest accuracy. We then chose H1 and H4 as the two highest-efficiency compute heads.

6.4.6 CAS - Determining the timing of context detection and initial branch

Next, we explored how the choice of placing the context detector at the early exit-points of the energy-efficient compute heads (H1 and H4) impacts RealGIN-MH performance.

As shown in table 6.2, the configuration '*Context @ Shufflenet*' (RealGIN-MH) achieves the highest accuracy, latency, and energy efficiency. This indicates that Shufflenet features are effective in making an accurate enough context determination, in-time. On the other hand, the configuration '*Context @ LSTM*', which relies solely on LSTM features from the audio data, achieves significantly lower accuracy, primarily for task instances where pointing input is important. This is expected since it is very likely that the verbal instruction ends much later than the corresponding pointing gesture, at which point it is too late to trigger the RGB and depth sensor to capture the pointing hand. This suggests that even though the audio sensor consumes the least energy, it is not suitable as a detector of task context. This example also illustrates that the typical sensor triggering approach (relying on a low-energy sensor to determine the

Table 6.2: Performance variations with context detectors added at various branch points

Context @	Accuracy (%)	Latency (ms)	Energy (mJ)		
			Processing	Sensing	Total
Shufflenet (H1)	**76.46**	**130**	**710**	**130**	**840**
LSTM (H2)	59.23	138	740	145	885
LSTM + Shufflenet (H21)	75.29	145	775	145	920
AFS (H3)	74.19	147	800	140	940
L-GARAN (H4)	73.56	150	820	135	955

context to trigger the high-energy sensors) is not appropriate or our multi-modal instruction comprehension task, which includes significant asynchronous input. The configuration '*Context @ LSTM+Shufflenet*', which combines language and visual features, offers a potentially better feature representation for context determination. However, it comes at the cost of higher overall latency and energy consumption, with a lower accuracy compared to '*Context @ Shufflenet*'. This loss in performance can be attributed to the increased delay in the decision to activate the depth camera, leading to higher pixel errors according to Figure 6.4(a). Thus, from Table 6.2, we can decide that our initial committed processing branch would be H1, which uses the RGB camera data.

6.4.7 RealGIN-MH - Multiple output heads

As depicted in figure 6.6, H2, H21 and H3 marked in purple are the redundant compute heads based on CAS-based optimal head selection approach. Thus, we remove these redundant heads and only activate H1, H4 and H5 for this step to subsequently train the context detector. Each output head provides a bounding box of the target object via Feature Pyramid Network (FPN) and regression. Let us assume, I_t as the key frame (RGB), L as the text instruction and G as the depth frames captured after the sensor is triggered. Let F_{out} be the final feature maps for bounding box regression. At runtime, we dynamically choose a specific compute head based on the context estimated by our context detector module.

To predict the task context, we use the visual features generated by the Shufflenet backbone. We further add a 2-layer CNN network followed by a single fully connected layer to compute

the feature embedding necessary to predict the visual complexity. This feature embedding is then sent to 3-neuron fully connected layer with Gumbell-Softmax activation function [47] to compute the discrete task context triple:

$$w_1, w_2, w_3 = G(F(f_v)); \quad where\ w_1, w_2, w_3 \in \{0, 1\} \tag{6.3}$$

As depicted in equation 6.3, we compute w_1, w_2, w_3 representing 3 distinct complexity levels, and the corresponding branches. When $w_1 = 1$ RealGIN-MH only uses a Shufflenet backbone for comprehension; when $w_2 = 1$, Shufflenet backbone for vision, Bi-LSTM for language and AFS gets activated, while $w_3 = 1$ implies the activation of all the modules (including the depth camera and the DDPM module).

Once the context is determined, we then define two forward computations in training and inference mode.

- **Training Mode** - In the training mode, to achieve differentiability during the back-propagation stage, we compute all three branches (regardless of the computed values w_1, w_2, w_3) as:

$$\begin{gathered} F_{H1} = H1(I_t); F_{H4} = H4(F_{H1}, L); F_{H5} = H5(F_{H4}, G) \\ F_{out} = w_1 * F_{H1} + w_2 * F_{H4} + w_3 * F_{H5} \end{gathered} \tag{6.4}$$

  Furthermore, we modify the original loss function of RealG(2)In-Lite l_{orig} as follows to add the policy for selecting the optimal compute head:

$$loss = l_{orig} + \frac{1}{N} * \sum_{i=0}^{N} (e_1 * w_1^i + e_2 * w_2^i + e_3 * w_3^i) \tag{6.5}$$

  Here, e_1, e_2 and e_3 are the relative energy costs for respective branch point and N is the batch size. Based on our energy profiling on Jetson TX2, we identified that the total energy for $H1$ is 561 mJ, $H4$ is 775 mJ and $H5$ is 10,915 mJ ($\sim$20x higher than $H1$). Thus we choose, $e_1 = 561/(561 + 775 + 10915) = 0.046$, $e_2 = 775/(561 + 775 + 10915) = 0.063$ and $e_3 = 10915/(561 + 775 + 10915) = 0.89$.

- **Inference Mode** - In the inference mode, to achieve savings in latency we compute *only* the relevant branch based on the task complexity.

$$
\begin{aligned}
if\ w_1 = 1 \rightarrow F_{out} = H1(I_t) \\
if\ w_2 = 1 \rightarrow F_{out} = H4(H1(I_t), L) \\
if\ w_3 = 1 \rightarrow F_{out} = H5(H4(H1(I_t), L), G)
\end{aligned}
\tag{6.6}
$$

6.5 Results

We now present the performance of RealGIN-MH, as well as the different baselines, using the COSM2IC multi-modal instruction dataset. While the COSM2IC instructions were collected using a Microsoft Hololens smart-glass, we experimentally found that the device currently neither has the computational resources to support the baseline models, nor is it straightforward to directly measure the energy consumed when toggling the depth sensor. Therefore, we used an alternate hardware setup, where the models are executed on an NVIDIA Jetson TX2 device [8], a pervasive device with greater programmability but resources in fact inferior to top-of-the-line commercial smartphones. The TX2 interfaces with a RealSense L515 depth sensor that can be easily toggled On/Off (via software commands), with power being measured accurately using a Monsoon power monitor [3]. Jetson TX2 also runs a real-time speech-to-text model called Picovoice cheetah [2] which converts the audio stream into text. We thus use an experimental setup where the audio and RGB camera data, corresponding to COSM2IC's environmental setup and verbal instructions, are captured on the HoleLens and then streamed to the nearby Jetson TX, where the depth sensor-augmented sensor data is processed by the RealGIN-MH and comparator models.

6.5.1 Evaluation Metrics

Similar to COSM2IC, we assume that the comprehension task is successful if the mid-point of the predicted bounding box lies within the ground-truth target object boundary. We measure

the depth sensing energy separately, as the other sensors are always on and thus have a constant energy consumption across all approaches. Since we observed that the L515 sensor consumes 1.5W of static power, we only measure the additional dynamic power consumed when the sensor is triggered to stream depth frames. For a comparison of energy consumption (Tables 6.3 and 6.4), we use the *average energy* consumed over all the instructions in the COSM2IC dataset–i.e., total energy consumed for the entire set of instructions, divided by the number of instructions in the dataset.

6.5.2 RealGIN-MH Performance against other baselines

Table 6.3 summarizes the performance of RealGIN-MH against various other baselines. For both RealG(2)In-Lite (end-to-end neural network approach) and COSM2IC (branch switching approach), the depth sensor is assumed to be on always-on, thereby consuming $\sim$10,000mJ energy/instruction. While COSM2IC optimizes the inferencing overhead, this translates to only 5.5% savings in the total energy cost.

In contrast, our proposed CAS-based RealGIN-MH jointly reduces both processing energy by $\sim$22% (by often using cheaper DNN branches) and sensing energy by $\sim$98.4% with respect to the RealG(2)In-Lite. In total, RealGIN-MH achieves $\sim$**12.9x** savings in total energy in comparison to RealG(2)In-Lite while maintaining a similar latency and suffering $< 2\%$ loss in task accuracy.

As a form of ablation study, we also evaluated another variant of RealGIN-MH, called RealGIN-MH-noDDPM, which dispenses with keyframe reconstruction and directly uses the delayed depth frame for pointing resolution. Due to the absence of this additional module, RealGIN-MH-noDDPM incurs $\sim$10% lower latency and consumes $\sim$10% (80mJ) lower energy than RealGIN-MH. However, the absence of DDPM degrades the task accuracy by an additional $\sim$2.5%.

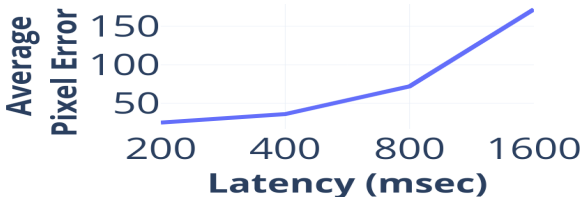
Table 6.3: RealGIN-MH performance against baselines

Model	Acc. (%)	Lat. (ms)	Energy (mJ)		
			Processing	Sensing	Total
RealG(2)In-Lite	78.51	155	853	10000	10853
COSM2IC	76.13	110	590	10000	10590
RealGIN-MH-noDDPM	73.97	115	630	135	745
RealGIN-MH	**76.46**	**130**	**710**	**130**	**840**

Table 6.4: Perf. of RealGIN-MH at various heads

Head	Acc. (%)	Lat. (ms)	Energy (mJ)	
			Proc.	Sens.
H1	68.51	102	561	0
H4	73.18	141	775	0
H5	78.20	165	915	10000
MH	76.46	130	710	130

(a) Startup Delay



(b) No. of DDPM frames

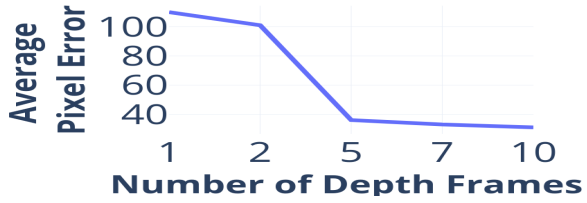


Figure 6.8: Average pixel error with DDPM

6.5.3 Branch-specific performance of RealGIN-MH

Table 6.4 provides the performance of RealGIN-MH, separately when all the instructions were forced to pass through a particular head. We see that while H1 (RGB only) or H4 (RGB + Audio) do not consume any depth-sensing energy, they both suffer from a significant degradation in task accuracy. In comparison, solely using the branch H5 with always-on depth sensing results in a superior accuracy of 78.20% while consuming a higher sensing energy consumption of 10000 mJ. RealGIN-MH dynamically chooses these branch points based on a complexity assessment, executing heads H1, H4 and H5 for 25.75%, 23.16% and 51.08% of the total instructions, respectively. Consequently, RealGIN-MH achieves task accuracy (76.43%) which is comparable to RealGIN-H5, but with a much lower average lower sensing energy of 130mJ.

6.5.4 Pointing sensitivity analysis

In Figure 6.8, we plot and observe how the average pointing error (in pixel distance) increases as the sensor activation delay increases. Thus, equipping future pervasive devices with faster sensor triggering capabilities may enable more accurate pointing resolution and higher task accuracy, and also obviate the need of the DDPM module for frame reconstruction. By varying the number of frames (N) used as an input to the DDPM, we observe (Figure 6.8) that using a larger number of frames results in a lower pointing error, but increases the sensing energy overhead. We empirically chose $N = 5$ frames, as additional frames provide only a negligible reduction in the pointing error.

Figure 6.7(b) and 6.7(c) visually illustrate (a) the 'keyframe' depth image–i.e., the depth image that we would have ideally used if the sensor was always on, and (b) the depth image regenerated using DDPM $N = 5$ frames. While the regeneration is not perfect, the pointing resolution is evidently adequate for the attention boosting performed by the G-GARAN module in RealGIN-MH.

6.6 Generalizability of CAS

Our results have demonstrated how CAS achieves a significant reduction in both sensing and processing for our canonical human instruction comprehension (target recognition) task. However, one may question whether CAS is generalizable to other tasks. For this, we consider two additional examples: a) Multi-modal human activity recognition (HAR), and b) Multi-modal image segmentation.

6.6.1 Multi-modal HAR

To illustrate CAS for human activity recognition, we leverage the *50Salads* dataset proposed by Stein et al. [92], designed for identifying complex gestures performed by different individuals

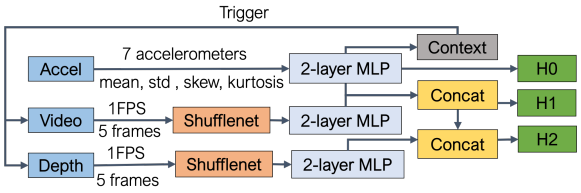


Figure 6.9: Architecture: Activity-MH pipeline

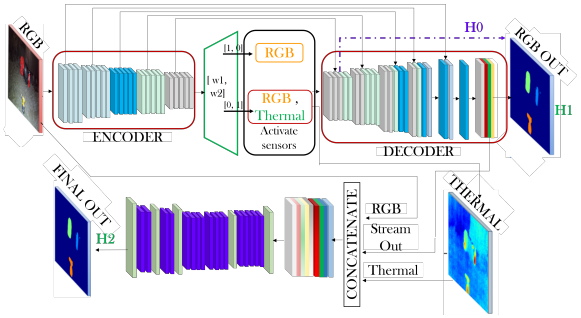


Figure 6.10: Architecture of PSTNet-Thermal-MH

while preparing mixed salads. The dataset captures these interactions through 7 accelerometer sensors attached to various kitchen objects, as well as an RGB and a depth camera positioned overhead.

Following the CAS paradigm, we introduce Activity-MH, an *on-device real-time model* illustrated in Figure 6.9, comprising three processing branches, namely accelerometer-based (Accel-based, H0), Accel+RGB-based (H1), and Accel+RGB+Depth-based (H2) activity classification. Each processing head consists of a fully-connected layer with an output size of 10 (corresponding to the 10 class labels).

Table 6.5: Head-based efficiency on 50Salads dataset

Branch	**Cost**	**Accuracy**	**Efficiency**
H0	0.018	0.46	2528.33
H1	1.45	0.57	39.5
H2	5.78	0.59	10.26

Through iterative training, we determined the accuracies and efficiencies of these heads, as summarized in Table 6.5. Subsequently, we trained the dynamic triggering approach (*Activity-MH*) using all three heads. Given the accelerometers have significantly lower power consumption (compared to RGB and depth camera), we propose to estimate the task context using the accelerometer-only pipeline *H*0. Table 6.6 presents the individual performance of H0, H1, H2 in the context of *Activity-MH* with the CAS paradigm. As illustrated, Activity-H2 outperforms others in accuracy, albeit at the expense of a high overall energy consumption of 5780mJ. On the other hand, Activity-H0 supports low-power activity recognition but suffers from reduced accuracy. Notably, *Activity-MH*, with its adaptive sense-making, consumes **5.96x** less overall

Table 6.6: Activity-MH performance against baselines

Model	Acc. (%)	Lat. (ms)	Energy (mJ)		
			Proc.	Sens.	Total
Activity-H0	45.51	10	18	0	18
Activity-H1	57.28	95	450	1000	1450
Activity-H2	59.29	155	780	5000	5780
Activity-MH	**58.10**	**42**	**190**	**780**	**970**

energy, operates 3.69x faster than Activity-H2, and incurs only a 1.10% accuracy loss.

6.6.2 Multi-modal Image Segmentation

We also applied CAS to a multi-modal segmentation task proposed in [89]. This model, called PSTNet-Thermal, is a multi-modal DNN that takes an RGB image and a thermal image as inputs and produces a segmentation output with 5 different classes.

Following the CAS principle, we added $N = 3$ heads to the PSTNet-Thermal, as illustrated in Figure 6.10. Through iterative training, we determined the accuracies and efficiencies of these heads, which are summarized in Table 6.7. Based on our first principles, we then selected H1 and H2 for our dynamic, multi-head model called *PSTNet-Thermal-MH*, which was trained using the dynamic triggering approach.

Table 6.7: Head-based accuracy on PST900 dataset

Branch	Cost	Accuracy	Efficiency
H0	0.03	0.46	15.33
H1	**0.04**	**0.67**	**16.75**
H2	**0.23**	**0.69**	**3**

PSTNet-Thermal-MH seeks to intelligently trigger the power-hungry thermal camera, which consumes 2.5W as per its technical specs) using a context detector placed at the encoder as shown in Figure 6.10. Table 6.8 plots the resulting accuracy and energy overheads, using spec-based power values (thermal= 2.5W, RGB=0.5W), of *PSTNet-Thermal-MH* vs. alternative baselines. We observed that PSTNet-Thermal-MH activates the thermal camera only 36% of the time, and achieves a **2x** reduction in total energy consumption, compared to PSTNet-Thermal, without any accuracy loss.

Table 6.8: PSTNet-Therm-MH performance Vs baselines

Model	Acc. (mIoU)	Lat. (ms)	Energy (mJ)		
			Proc.	Sens.	Total
PSTNet	0.6765	20	50	5	55
PSTNet-Thermal	0.6837	45	121.50	123.75	245.25
PSTNet-Therm-MH	**0.6822**	**31**	**83.7**	**32.86**	**116.46**

6.7 Discussion

CAS represents a significant advancement by addressing a key issue overlooked in COSM2IC, namely considering sensing energy overhead and proposing a joint dynamic model optimization to mitigate both processing and sensing energy overheads. With CAS achieving approximately a 12.6x reduction in overall energy overheads, we envision several future directions to further enhance its performance.

- **Hardware Triggering:** Our RealGIN-MH implementation only utilized software-based activation of the depth sensor. We observed a static power consumption of $\sim$1.5W even when the depth sensor is presumably in a low-power *standby* state. Additional energy savings can be realized by introducing a hardware switch and supporting much faster ($\leq$100 msecs) sensor activation. We believe that our CAS paradigm, enabled by such fast on/off depth sensor switching, will be key to assuring energy efficiency for a suite of emerging spatial computing applications (which require depth sensor input) and urge hardware manufacturers to support this capability.

- **Improving DDPM Energy Efficiency:** Our current DDPM module, where a past depth frame is estimated using only a series of other depth frames, currently consumes non-trivial energy. For further energy optimization, it may be possible to perform depth image synthesis, using approaches such as Wofk et al. [110], from the already-available RGB frames sharing the same viewpoint. We could also consider stereo vision cameras (where CAS is used to selectively invoke the second camera) to replace expensive depth sensors.

6.7.1 Achieved Design Goals

Building on previous work that addressed all three design goals—multi-modal sense-making, low-latency sense-making, and battery-powered sense-making—CAS aims to reduce the sensing energy overhead resulting from the additional sensors, such as LiDAR, used for capturing pointing gestures. While COSM2IC did not address the sensing energy overhead, evaluations in this chapter have shown that sensing energy accounts for nearly 94% of the overall energy consumption of the COSM2IC model. Therefore, extending the proposed dynamic model optimizations is crucial for increasing operating time when executing these models on a battery-powered pervasive device.

Similar to previous chapters, RealGIN-MH and other baseline models achieved latencies well below the desired 1 second, ensuring interactivity and responsiveness. Extending from this, RealGIN-MH achieved a 12.93x reduction in overall energy overhead while maintaining task accuracy within the desired range of $\sim 5\%$ making a significant contribution to the design goal of battery-powered sense-making.

6.8 Summary

In the previous chapter, we introduced a dynamic optimization approach termed COSM2IC for Real-time and *efficient* (i.e., low energy overhead, high accuracy) comprehension of human instructions. While COSM2IC enables the multi-modal REC model to execute with $\sim$4x reduction in processing energy overheads, it fails to consider the sensing energy overheads. Thus, we have introduced the CAS paradigm that simultaneously reduces both the energy and inferencing overheads for such complex inferencing tasks. Key to this approach is utilizing the depth camera solely as an alternative sensor for capturing visual context and pointing gestures, activating it only when necessary through an dynamic execution pipeline. In summary, we have made the following key contributions with CAS framework.

- *CAS-Based, Multi-Branch Multi-modal Instruction Comprehension Model*: We introduce and develop the CAS paradigm, where the DNN-based model consists of multiple branches (each with independent heads) of varying complexity, which can be executed selectively. We then show how an *efficiency* metric, embodying accuracy vs. energy tradeoffs, can be used to judiciously identify one of the branches as the basis for early task context estimation and subsequent branch switching. We exemplify CAS by adopting and modifying RealG(2)In-Lite [107] to develop RealGIN-MH, a multi-branch model for multi-modal target object acquisition. Given a set of three possible sensor combinations (RGB camera alone, <RGB cam+audio> and <RGB cam+audio+depth>) RealGIN-MH initially commits to a branch that uses RGB camera data alone, while its integrated complexity detector uses features from the visual backbone to switch to other branches if needed, triggering the energy-intensive depth camera only when warranted.

- *Reconstruction of depth image keyframe*: While triggered activation of the depth sensor provides significant energy savings, we experimentally observe that its non-negligible (380-420 msec) activation latency often implies that the sensor is activated after its interval of relevance, when the user actually points to the target object, has elapsed. To overcome the resulting 3-4% loss in comprehension accuracy, we utilize a lightweight shallow neural network to *regenerate* the user's unobserved pointing gesture, with a median error of only 36 pixels, using a stack of $N = 5$ future depth frames.

- *Demonstration of Performance Benefits*: We implemented RealGIN-MH on a Nvidia Jetson TX2 platform, integrating the pervasive device with a representative RGB camera, microphone and Intel RealSense L515 LIDAR depth camera. We replay the sensor data captured in the COSM2IC dataset (220 tasks over 64 distinct scenes of varying scene and instructional complexity) and empirically show that RealGIN-MH can achieve a $\sim$**12.6x** reduction in energy overheads, with comparable accuracy, compared to both (a) RealG(2)In-Lite (a static optimized version of RealGIN suitable for Jetson devices) and(b) COSM2IC, a baseline that optimizes inferencing but not sensing.

- *Demonstration of Generalizability*: We demonstrate the generalizability of CAS using two additional multi-modal task: (a) semantic segmentation of simultaneously acquired RGB and thermal camera images, and (b) multi-modal recognition of cooking actions. For segmentation, we apply CAS to develop a new multi-headed *PSTNet-Thermal-MH* model, which activates the power-hungry thermal camera only 36% of the time, achieving $>$ 50% energy savings over a baseline CNN-based embedded PSTNet-Thermal [89] model. For cooking action recognition, we apply CAS to develop *Activity-MH* which takes accelerometer sensor data and activates RGB camera only 16.67% and depth camera only 12.21% of the time. This achieves 5.96x savings over a baseline multi-modal activity recognition model.

Part II

Video-based Human Instruction Comprehension

Chapter 7

Efficient Video Grounding with Dynamic Frame Skipping

In the preceding chapters, we have extensively delved into the challenges of comprehending multi-modal human instructions. Various approaches have been explored, employing diverse sensing modalities and optimization techniques to achieve real-time and efficient comprehension of instructions. However, these methodologies have predominantly focused on utilizing a single static image as the visual input.

Nevertheless, we contend that not every instruction comprehension scenario can be adequately addressed by relying solely on a single static image, but a series of image frames or videos. Put simply, instructions that cannot be fully comprehended through a single static image belong to a more intricate category of human instructions. Our main goal is to investigate whether the dynamic model optimizations we propose remain effective in handling these complex scenarios. To demonstrate this, in this chapter we delve in to video-based instruction comprehension, enhanced through dynamic model optimization technique to enable real-time and efficient instruction comprehension with videos.

Figure 7.1: Example scenario for video-based human instruction comprehension

7.1 Introduction

To illustrate video-based comprehension, consider the scenario depicted in Figure 7.1. Here, a worker instructs one robot to pick up a box after another robot completes unloading it. In such instances, the grounding model needs to encompass not only spatial understanding but also temporal comprehension to identify the precise time window for picking up the specified box.

Moreover, the complexity of this instruction involves the worker pointing at the robot, shelf, and the target object to be picked. Thus, such instructions cannot be adequately characterized by a single image; instead, they necessitate the incorporation of video frames. To successfully address these complex instructions, a grounding model capable of both spatial and temporal understanding becomes essential.

In AI literature this task is termed as Spatio-Temporal Video Grounding (STVG). More concretely, STVG is a compound task that combines both spatial and temporal grounding of a natural language query to identify (a) the target object/region-of-interest and (b) the temporal segment where the target appears, in a given video. Thus the output of this multi-modal

STVG task is often visualized as a spatio-temporal tube extracted from the video by identifying the target object over several consecutive frames of the relevant video segment. Such a grounding is fundamental for video understanding and STVG is significantly more challenging than related tasks such as referring expression comprehension [129, 115, 32, 121] or temporal video grounding [16, 37] as STVG models need to be able to perform both temporal and spatial reasoning over multiple video frames. Generally, grounding involves the use of cross-modal attention mechanisms to fuse information from both language and video inputs. Since STVG models require such cross-modal attention mechanisms to be repeated across several frames of a video, they end up being computationally intensive. The STVG problem was introduced by Zhang et al. [126] along with a supporting dataset called VidSTG. Prior to this work, the problems of spatial grounding and temporal grounding were treated separately such that the STVG task could be addressed by performing precise temporal grounding (identifying the relevant video segment) followed by spatial grounding on the identified segment. This is obviously inefficient and Zhang et al.,[126] showed that models such as STGRN [126] that perform the compound task of STVG simultaneously provide significant performance benefits. Today, transformer based models such as TubeDETR[114], STVGBert [93], STCAT [48] dominate the performance charts in this challenging vision-language task. This is expected, given the success of transformer based models [58, 95, 69, 94] in several vision-language tasks such as image-text retrieval, caption generation etc.

Among the transformer based models, two-stage approaches (e.g., STGRN) that rely on pre-generated object/tubelet proposals are less efficient compared to single-stage approaches (TubeDETR and STCAT) that parse the video only once through their pipeline. These two-stage models exhibit higher computational complexity and latency for a similar accuracy compared to single-stage approaches. Even in such single-stage models, the repeated use of transformer based encoder-decoder pairs continues to incur high computational overhead. They also require a large memory as they load the entire videos and execute the compute-heavy transformer encoders on every single frame to retain the resulting intermediate feature maps. The intermediate attention-based features obtained from the decoder are used by prediction heads to generate the following three outputs for every single video frame— (a) a bounding box, (b) a probability

that this frame is the start-frame of a tube and (c) the probability that this frame is the end-frame of a tube. Based on our evaluation of TubeDETR [114] and STCAT [48], processing latency steadily increases with the video length (see Figure 8.5). It is thus crucial to reduce the computational footprint of STVG models, especially for practical use-cases that (a) involve long untrimmed videos (e.g., security footage or instructional videos), or (b) weave such grounding into interactive applications (e.g., having service bots respond to human assistance requests).

We thus seek to develop models that can perform STVG with reduced runtime overhead and latency. On a closer re-look at the state-of-the-art (SOTA) approaches, we first observed that it is often sufficient to execute the STVG pipeline on a small set of candidate moments/frames within a given video (e.g., when there is a scene change or when we see a frame that contains relevant objects or gestures mentioned in the accompanying text). We can thus save significant computation if we are able to identify such anchor moments (frames) efficiently, track the relevant visual changes within the temporal vicinity of such frames and skip the execution of the complex encoder-decoder blocks for the rest of the less-salient frames. We effectively desire to perform an imprecise, light-weight *temporal anchoring* before executing full-blown STVG models. This principle becomes the key to our proposed model for STVG, ***GRefExSel*** (**G**rounding **Ref**erring **Ex**pressions via Anchor Frame **Sel**ection), where an early estimator termed ***Adaptive Frame Selector (AFS)*** identifies whether a given frame is an anchor frame, using the features extracted from the linguistic expression as well as from the neighbouring frames within a time-window. The complex transformer encoder-decoder pipelines are then executed exclusively only on such anchor frames, saving significant computational resources. Apart from identifying the anchor frames, AFS also provides a feature representation (called delta features) of the visual changes between the current frame and the last identified anchor frame. These delta features can then be added to the decoder output (attention based features) of the previous anchor frame to generate an approximation of the attention-based features for each non-anchor frame.

7.2 Baseline

We identified STCAT [48] and TubeDETR [114] as the current state-of-the-art STVG models. Both of them use single-stage architectures with a) ROBERTa [66] model for language encoding, b) ResNet-101 model as visual feature backbone, c) Transformer encoder for calculating multi-modal video-text features, d) Transformer decoder for modelling temporal dependencies and d) 2 prediction heads for calculating spatio-temporal bounding box tubes. Despite their single-stage architecture, these networks are generally computationally intensive due to the following reasons.

1. These approaches consist of an expensive visual backbone of ResNet-101 [43] model and multiple transformer encoder decoder blocks for modelling multi-modal (video + text) features as well as the temporal dependencies.

2. These networks demand all the frames of the input video (partially or fully) to pass through a resource-intensive pipeline involving multiple transformer blocks. TubeDETR[114] identified that this is very computationally intensive and used a fast-slow two stream approach in their encoder that samples the video uniformly such that only one out of every k frames pass through the transformer encoder(slow multi-modal stream) to get the attention based features. However, TubeDETR still demands all the frames to pass through a complex RESNET backbone and a transformer decoder. On the other hand, STCAT, which achieves a higher accuracy demands all of the frames to pass through its complete pipeline including the encoder and decoder. Such computation, demands an average latency of 30msec per image and memory of 1.7GB for TubeDETR and 48msec per image and memory of 2.3GB STCAT on average for VidSTG dataset even on a server-grade computer.

To overcome these limitations, we looked into suitable approaches for reducing the computational load involved in spatio-temporal video grounding. While approaches like model pruning and skipping [104] have been studied for unimodal tasks, they are not easy to apply for net-

works used in multi-modal tasks such as STVG. In the prior chapters where we introduced SoftSkip and COSM2IC, we have highlighted this problem and have come up with specialized approaches for multi-modal tasks such as referring expression comprehension. Even so, such optimization has not been attempted for transformer based architectures that are common in STVG. However, we observed that the main computational load comes from the fact that all the frames need to pass through the whole pipeline. With this in mind, we attempted to develop our FrameSkip approach.

7.3 FrameSkip Approach

We now describe the FrameSkip approach proposed in this chapter. On a closer look at the STVG datasets and models, we observed that it is unnecessary to execute the entire grounding pipeline on all the frames. Instead, such execution can be limited to some key moments in the video, such as important scene changes or appearance or disappearance of relevant objects or actions. Hence, we will be able to reduce inference latency if we execute the compute-heavy transformer encoder-decoder blocks only on frames associated with such anchor moments. However, we need to consider two main challenges when adopting such an approach. First, the identification of such anchor frames should be computationally inexpensive, while still utilizing features from both visual and language inputs. Second, even if we can determine such anchor moments (frames), we cannot simply ignore the rest of the frames as they also contain significant spatiotemporal context. Specifically, the prediction heads of STVG models, that use the output of the decoder, expect that every frame has been processed by the encoder-decoder pair to model the cross-modal information as well as the spatiotemporal context. Hence, we need to devise a way to approximate the decoder output for such non-anchor frames as well. To achieve this we first need to obtain a feature representation of the visual differences between a given frame and the last identified anchor frame. If we have such a differential feature representation, intuitively, the encoder output of a non-anchor frame could be approximated by adding this differential representation to the encoder output of the last identified anchor frame.

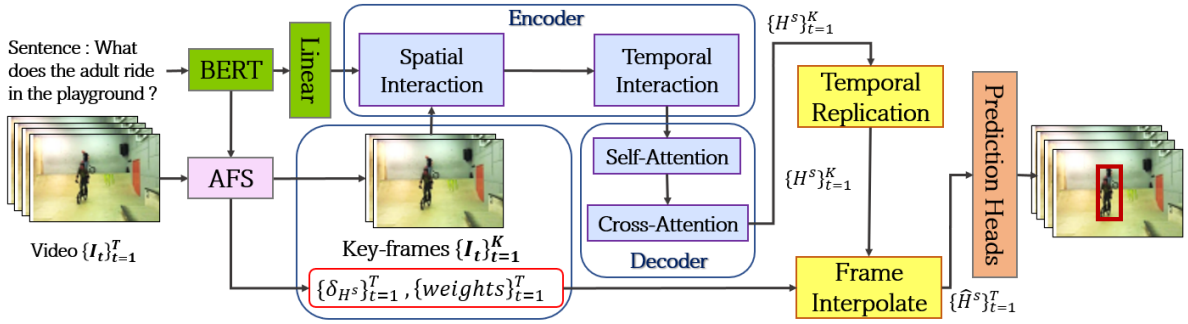


Figure 7.2: Overall Architecture of GRefExSel

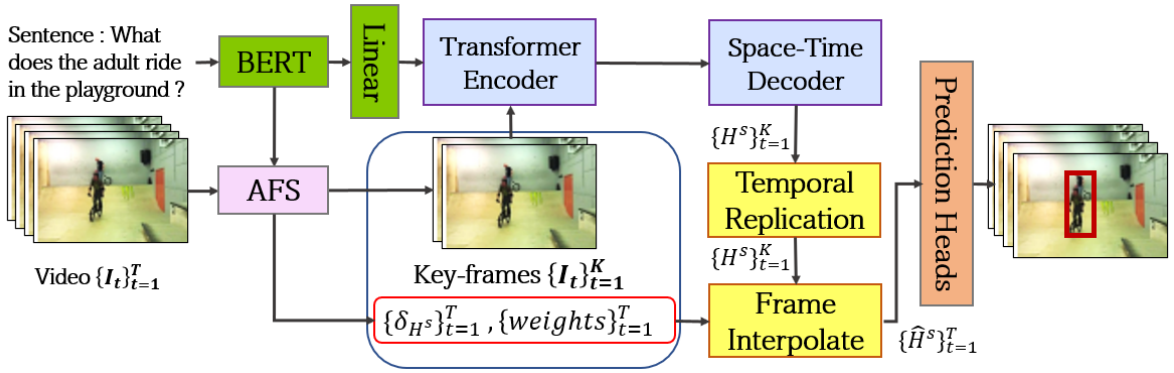


Figure 7.3: Overall Architecture of GRefExSel-lite

We next describe the AFS module that implements the ideas mentioned above.

7.3.1 Adaptive Frame Selector

As shown in Figure 7.4, AFS takes a series of video frames and language features as input. The video frames are passed through a single residual block of the ResNet-101 [43] backbone of TubeDETR. These features are converted to a 2-dimensional feature vector of size $\mathscr{R}^{T\times D}$, using a fully connected layer of output dimension of $D = 256$. To model the visual differences due

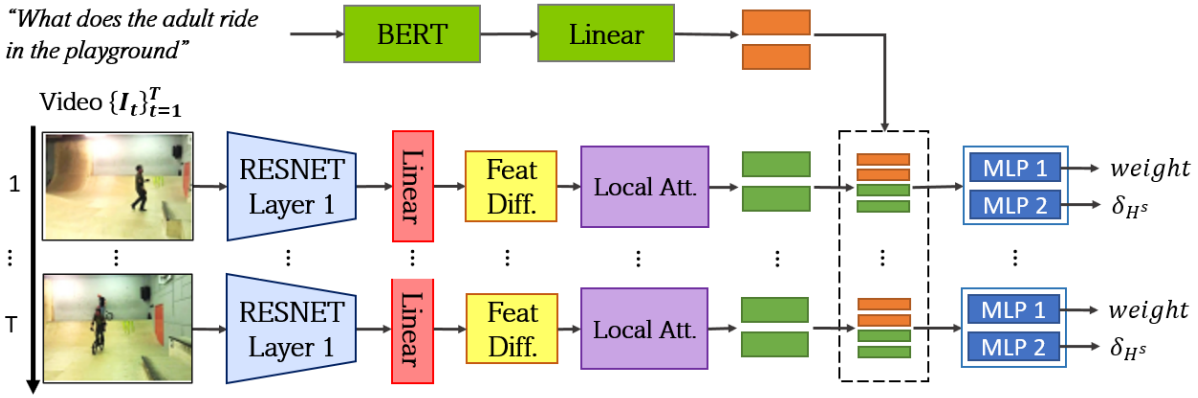


Figure 7.4: Adaptive Frame Selector Architecture

to scene changes, we take the difference between feature vectors from adjacent frames. These differences in feature vectors are then passed through a window-based local attention network [18] (window size=10) to model the temporal dependencies in the local neighbourhood to obtain $V_t \in \mathscr{R}^{T \times D}$. Each visual feature V_t is concatenated with the language features computed using a ROBERTa model. Finally, AFS uses 2 separate inexpensive Multi-Layer Perceptron (MLP) models with 3 layers and dropout for regularization (dropout probability=0.5). One of these MLPs is used to predict whether a given frame is an anchor-frame or not. The other MLP is used to obtain a feature representation of the visual differences between the current frame with respect to the nearest anchor-frame. We call this output as the *delta feature* representation. These delta features will then be used to interpolate the decoder output to obtain cross-modal attention-based features for non-anchor frames, so that they can used by the prediction heads of the STVG pipeline to predict the spatiotemporal tubes. The operation of the two MLPs can be expressed as given below.

$$\begin{aligned} weights = Gumbell(MLP[F_l, V_t]) \;\; where \;\; weights \in \mathscr{R}^{T \times 2} \\ \delta_{H^s} = MLP[F_l, V_t] \;\; where \;\; \delta_{H^s} \in \mathscr{R}^{T \times D} \end{aligned} \tag{7.1}$$

In the above equation, F_l refers to the language features taken from the same text encoder in TubeDETR, V_t refers to the visual features obtained after computing local attention for a video of length T frames, while *weights* determine whether the respective frame is an *anchor-frame* or a *redundant-frame*. We use Gumbell-Softmax activation function, which returns discrete 0 or 1 value for elements in *weights*. When $weights_i[0] = 1$ and $weights_i[1] = 0$, i^{th} frame is deemed to be a *anchor-frame*. When $weights_i[0] = 0$ and $weights_i[1] = 1$, i^{th} frame is deemed to be a *redundant-frame*. We also assume that the first frame of the video is always an anchor-frame. The output of the second MLP, denoted by δ_{H^s}, represents the delta features that were described earlier.

7.3.2 Loss Function

We propose a modification to the original loss function used in STVG models to reward the model for skipping more frames. At the training stage, we define a *frame-skip* parameter which is a hyper-parameter that determines the percentage of frames that are considered to be redundant in the video. A value closer to 1 would encourage aggressive skipping behaviour, thereby reducing computations. A value closer to 0 would encourage conservative skipping behaviour resulting higher computational load. Let this frame-skip parameter be denoted as S and original loss function of VG be $\mathscr{L}_{VG}$. We formulate the following reward loss in our FrameSkip approach.

$$\begin{aligned} \mathscr{L} = \mathscr{L}_{VG} + \lambda_{dIoU}\mathscr{L}_{dIoU}(\hat{b},b) + \frac{\lambda_w}{N}\sum_{i=0}^{N}|weights[0]_i - (1-S)| \\ where\ \ \mathscr{L}_{dIoU}(\hat{b},b) = (1 - dIoU(\hat{b},b)) \end{aligned} \tag{7.2}$$

Let us assume that t_s and t_e are the ground truth start and end times of the target object. Then, $b \in [0,1]^{4\times(t_e-t_s+1)}$ and $\hat{b} \in [0,1]^{4\times(t_e-t_s+1)}$ denotes the ground truth and predicted bounding box coordinates respectively for the duration where the target object appears. In addition to the original STVG loss function, the second and third terms in the loss function act as our reward function. The second term rewards the model for correct bounding box prediction using Distance IoU metric [128]. The third term, which is a Mean Absolute Error term rewards the model for skipping a desired fraction of frames as redundant frames. Since S refers to the desired average fraction of frames to be skipped, $1-S$ refers to the desired average fraction of anchor frames to be identified. λ_{dIoU} and λ_w are two hyper-parameters set for specifying the relative weights for the respective loss terms. These hyper-parameters are set to 1.0 and 2.0 respectively in the training stage.

Figures 7.2 and 7.3 summarize the FrameSkip paradigm, applied to a typical encoder and decoder based STVG pipeline to propose the two models, GRefExSel and GRefExSel-lite, that

we shall now describe. In particular, the central principle is to only send the *anchor-frames* to the expensive encoder-decoder blocks and use a late interpolation approach (explained in Equation 7.3) to obtain the corresponding feature representation for the redundant frames.

7.3.3 GRefExSel

We apply this FrameSkip framework to propose an optimized STVG model named GRefExSel. Figure 7.2 illustrates the architecture of GRefExSel. GRefExSel comprises the following components along with the AFS module explained earlier.

7.3.3.1 Language Encoder:

GRefExSel uses ROBERTa [66] as the language encoder. Last hidden state output of ROBERTa is followed by a Linear layer to obtain $F_l \in \mathscr{R}^{L \times d}$ language features where L is the word length and $d = 256$ is the hidden dimension. These language features are obtained at the AFS stage and reused here.

7.3.3.2 Visual and Multi-modal Encoder:

After the AFS step, we get a series of K key-frames, $\{I_t\}_{t=1}^{K}$. These K frames are passed to the ResNet101 [43] backbone to obtain $F_v \in \mathscr{R}^{K \times HW \times d}$. H and W are the resolutions of the feature maps from the ResNet backbone. Each $\{f_v\}_{i=1}^{M}$ is concatenated with F_l to obtain the multi-modal feature representation, which is then passed to 2 transformer encoders with 6 layers (similar to STCAT) comprising of the spatial interaction and temporal interaction layers. The key idea in the spatial interaction layer is to model spatial relationships in individual frames of the video, whereas the temporal interaction layer models the temporal dependencies across the video.

7.3.3.3 Multi-modal Decoder:

To model the temporal interactions among a set of video-text features, STCAT proposes a query-guided decoding network with dual-decoder architecture. We adopt this dual-decoder network as the multi-modal decoder which takes the output from the visual and multi-modal encoder. Multi-modal decoder outputs a feature representation $H^s \in \mathscr{R}^{K \times d}$.

7.3.3.4 Redundant Frame Interpolation:

As mentioned earlier, we always assume that the first frame of a video to be an anchor-frame. While an anchor frame is passed through the encoder-decoder pair to obtain attention based features, the subsequent redundant frames by-pass the encoder-decoder pair. In order to obtain the attention based features for such redundant frames, we adopt an interpolation based approach. First, to match the original sequence length of T video frames, we temporally replicate H_i^s till the next $(i+1)^{\text{th}}$ anchor frame is found. We repeat this for $i = 0$ to $i = K-1$ till we match the same sequence length of T. Then, for each of the temporally replicated frame H_s, we add (as shown in Equation 7.3) the corresponding delta features δ_{H^s} calculated during the AFS step to obtain $\hat{H}^s$.

$$\hat{H}^s = H^s + weights[1] * \delta_{H^s} \tag{7.3}$$

Intuitively, each of these interpolated frames approximates the attention based features of the corresponding redundant frame by combining the attention based features of the previous anchor-frame with the delta features that represent the visual differences of the redundant frame with respect to the anchor frame.

7.3.3.5 Prediction Heads:

We use two MLPs as prediction heads that take $\hat{H}^s$ as an input: (a) a 3-layer MLP for predicting bounding box coordinates, $\hat{b} \in [0,1]^{T \times 4}$ and (b) a 2-layer MLP for predicting start and end time

probabilities $\hat{\mathcal{T}}_s \in [0,1]^T$ and $\hat{\mathcal{T}}_e \in [0,1]^T$. During inference, the start and end times of the target object $\hat{t}_s$ and $\hat{t}_e$ are computed by taking the maximum of joint start and end time probabilities with invalid combinations masked out (i.e., $\hat{t}_s \geq \hat{t}_e$)

GRefExSel is trained for a single epoch with the initial weights taken from trained STCAT model and the loss function as explained in Equation 6.5, where $\mathscr{L}_{VG}$ is replaced with the original loss function of STCAT and frame-skip parameter S is set to 0.7.

7.3.4 GRefExSel-lite

The encoder and decoder structure used in GRefExSel (adopted from STCAT), was found to be comparatively computationally intensive. To further reduce the computational load, we propose an alternative model named GRefExSel-lite. In GRefExSel-lite, the AFS, language encoder, RESNET visual backbone, redundant frame interpolation and prediction heads remain identical to GRefExSel. Motivated from TubeDETR, we however adopt a single transformer encoder for multi-modal features, and the space-time decoder as proposed in TubeDETR for decoder network as shown in Figure 7.3. GRefExSel-lite is trained for a single epoch with the initial weights taken from the trained TubeDETR model and the loss function is modified as explained in Equation 6.5, where $\mathscr{L}_{VG}$ is replaced with the original loss function of TubeDETR and frame-skip parameter S is set to 0.7.

7.4 Results

We now experimentally evaluate GRefExSel and GRefExSel-lite vs. alternative state-of-the-art VG models. In general, we follow the same evaluation methods utilized in prior work [114, 48]. In addition to the standard metrics, we will also consider latency, frames per second (FPS) and memory consumption as additional performance metrics.

7.4.1 Datasets and Evaluation Metrics

Datasets: To evaluate the performance of the models, we follow the previous works to use VidSTG [126] and HC-STVG [97] as the benchmark datasets for STVG. Both datasets are annotated with spatio-temporal tubes with start, end frames and bounding boxes for each frame corresponding to text queries. **VidSTG** consists of 99,943 sentence descriptions with 44,808 declarative sentences and 55,135 interrogative sentences describing 79 types of objects appearing in 10,303 different videos. The dataset is divided into training, validation and test subsets with 80,684, 8,956 and 10,303 distinct sentences respectively, and 5,436, 602 and 732 distinct videos respectively. videos. **HC-STVG** is a human-centric STVG dataset with 5,660 videos in multi-person scenes, each associated with one language expression that is related to human attributes or actions. HC-STVG is divided into the training set and the testing set with 4,500 and 1,160 video-sentence pairs, respectively.

Evaluation Metrics: We follow [126] and use m_vIoU, m_tIoU and $vIoU@R$ as accuracy metrics. $v\_IoU = \frac{1}{|S_u|}\sum_{t\in S_i} IoU(\hat{b}_t, b_t)$, where S_u and S_i refers to the intersection and union between the predicted tube and ground truth tube. $IoU(\hat{b}_t, b_t)$ refers to the IoU score between the predicted bounding box and target bounding box at the frame t. Thus, m_vIoU refers to the mean $vIoU$ score across the dataset and $vIoU@R$ refers to the percentage number of predictions achieving a $vIoU$ score of at least R. $tIoU$ measures the temporal overlap between the predicted and ground truth tubes and is defined as $tIoU = \frac{|S_i|}{|S_u|}$. m_tIoU then refers to the mean $tIoU$ across the dataset. In addition, we measure the system-level metrics (memory and inference latency) using a server with two Tesla T4 GPUs, each with a memory of 15GB.

7.4.2 Performance on Benchmark Datasets

Tables 7.1 and 7.2 summarize the performance of the proposed models against benchmark STVG models on VidSTG and HC-STVG datasets respectively. From our experiments, we observe that the FrameSkip mechanism allows GRefExSel and GRefExSel-lite models to select

Table 7.1: Performance comparisons of the state-of-the-art on the VidSTG test set (%)

Methods	Declarative Sentences					Interrogative Sentences				
	m_tIoU	m_vIoU	vIoU@0.3	vIoU@0.5	% frames skipped	m_tIoU	m_vIoU	vIoU@0.3	vIoU@0.5	% frames skipped
Factorized:										
GroundeR [85]+ TALL [37]		9.78	11.04	4.09	0		9.32	11.39	3.24	0
STPR [113] + TALL [37]	34.63	10.40	12.38	4,27	0	33.73	9.98	11.74	4.36	0
WSSTG [28] + TALL [37]		11.36	14.63	5.91	0		10.65	13.90	5.32	0
GroundeR [85] + L-Net [25]		11.89	15.32	5.45	0		11.05	14.28	5.11	0
STPR [113] + L-Net [25]	40.86	12.93	16.27	5.68	0	39.79	11.94	14.73	5.27	0
WSSTG [28] + L-Net [25]		14.45	18.00	7.89	0		13.36	17.39	7.06	0
Two-Stage:										
STGRN [126]	48.47	19.75	25.77	14.60	0	46.98	18.32	21.10	12.83	0
STGVT [97]		21.62	29.80	18.94	0					0
OMRN [125]	50.73	23.11	32.61	16.42	0	49.19	20.63	28.35	14.11	0
One-Stage:										
STVGBert [93]		23.97	30.91	18.39	0		22.51	25.97	15.95	0
TubeDETR [114]	48.10	30.40	42.50	28.20	0	46.90	25.70	35.70	23.20	0
GRefExSel-lite (Ours)	**47.50**	**29.10**	**41.10**	**27.34**	**66**	**45.76**	**24.23**	**34.23**	**22.11**	**67**
STCAT [48]	50.82	33.14	46.20	32.58	0	49.67	28.22	39.24	26.63	0
GRefExSel (Ours)	**49.19**	**32.36**	**45.05**	**31.57**	**64**	**49.02**	**27.27**	**38.51**	**25.77**	**61**

just about 35% of the video frames as anchor frames, enabling the remaining frames to by-pass the encoder-decoder blocks. However, this results in a minor drop in their accuracy when compared to STCAT. The m_vIoU of GRefExSel is just 0.55% shy of STCAT on the human-centric HC-STVG dataset. On the more general VidSTG dataset, GRefExSel achieves an m_vIoU that is just about 0.95% short of STCAT for the interrogative sentences and about 0.78% short of STCAT for the declarative sentences. Compared to the TubeDETR, GRefExSel is able to achieve higher m_vIoU on all the benchmark tasks and also achieve multi-fold improvement in grounding speed. Overall, we believe that this drop in accuracy is tolerable since nearly 63% of the frames are identified and selected to bypass the transformer encoder-decoders. We believe that this will translate into a significant savings in latency, energy and memory requirements when dealing with longer untrimmed videos in practical use-cases.

We then measured the average FPS and memory consumption of GRefExSel and GRefExSel-lite and compared them against TubeDETR and STCAT models in Figures 7.5 and 7.7. We found that GRefExSel runs 4.32x faster while consuming 24.78% less memory compared to STCAT while GrefExSel-lite is 5.18 times faster than STCAT while consuming 43.04% less memory. In Figure 7.6, we plot the latency of TubeDETR and STCAT against GRefExSel models and observe that the latency of each of these models increases steadily with the number of frames. However, the rate of increase of latency for GRefExSel and GRefExSel-lite is significantly lower than that of TubeDETR and STCAT.

Table 7.2: Performance comparisons of the state-of-the-art on the HC-STVG test set (%)

Methods	m_tIoU	m_vIoU	vIoU @0.3	vIoU @0.5	% frames skipped
Two-Stage:					
STVGT	-	18.15	26.81	9.48	0
One-Stage:					
STVGBert	-	20.42	29.37	11.31	0
TubeDETR	43.70	32.40	49.80	23.50	0
GRefExSel-lite	**43.12**	**31.95**	**49.01**	**22.78**	**65**
STCAT	49.44	35.09	57.67	30.09	0
GRefExSel	**48.89**	**34.23**	**56.19**	**29.26**	**63**

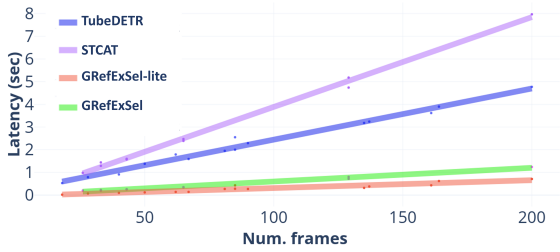


Figure 7.6: Latency vs number of video frames for GRefExSel and GRefExSel-lite against other VG models

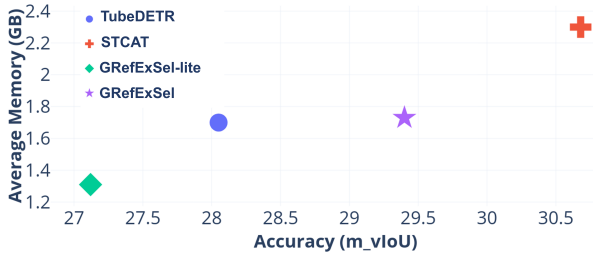


Figure 7.7: Comparison of memory consumption of GRefExSel and GRefExSel-lite against other VG models

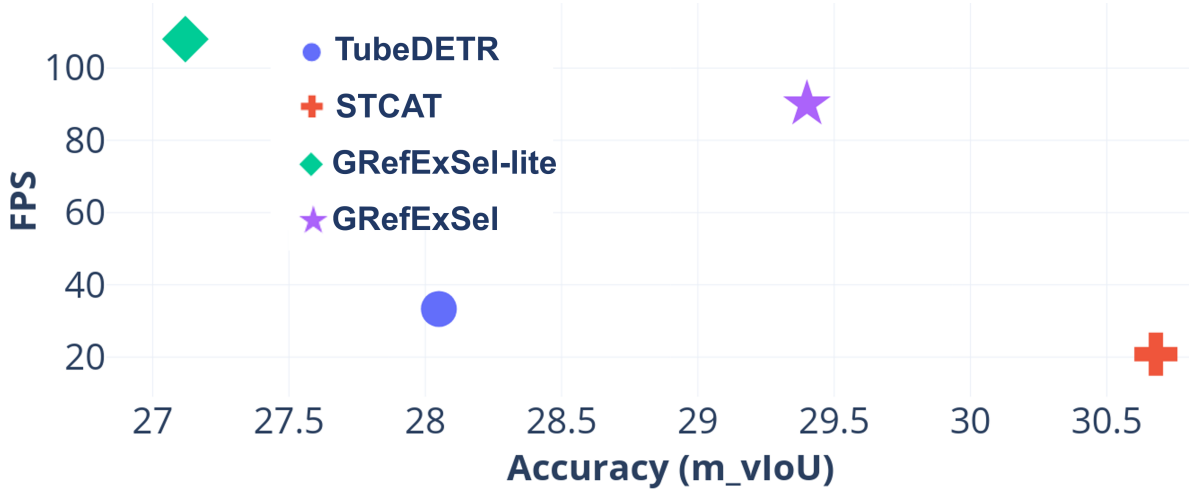


Figure 7.5: Comparison of FPS of GRefExSel and GRefExSel-lite against other VG models

Figure 7.8: Qualitative analysis of the functionality of AFS

7.4.3 Qualitative analysis of FrameSkip's skipping approach

We then looked at some of the instructions in the dataset to understand how the AFS module picks the anchor-frames and identified that it picks up anchor frames whenever (a) there are significant changes in visual content compared to adjacent frames, (b) visual content partially matching the words in the text and (c) target object appears in a frame.

Figure 7.8 shows one such example scenario taken from VidSTG dataset. This figure shows a series of frames that are identified as anchor-frames along with the starting frame for the given instruction. In the second frame, it appears that the scene has significantly changed, thus AFS captures this as a anchor-frame. In third frame, we can identify that a 'baby' appeared in the scene which exists as a key-word in the textual instruction. Finally, we can identify the correct target object appearing in the scene. Beyond this scene, we identified an increase in the number of anchor-frames detected.

7.5 Discussion

The GRefExSel model is primarily tailored for addressing the vision and language-based STVG task. However, as elucidated in our earlier chapters, natural human interaction extends beyond just language and visual cues to encompass other modalities such as gaze and hand gestures. Taking pointing gestures as an exemplar, a crucial assumption made during both COSM2IC and CAS is that the user may only point to a single target object. Yet, in reality, a user may point at multiple objects while issuing an instruction. Moreover, pointing gestures in such scenarios may not span the entirety of the video but instead be temporally confined to specific segments.

In these cases, an anchor frame could denote frames capturing the user steadily pointing at an object, furnishing significant information for STVG. Our AFS module could leverage this additional information to classify these frames as anchor frames.

Regarding gaze, a typical behaviour of a user issuing an instruction involves following the target object through their gaze. Hence, minor variations in gaze patterns could serve as additional delta features, akin to what was proposed in the GRefExSel approach. Integrating these insights can enrich the capabilities of our model, enabling it to better interpret and respond to multi-modal human interactions.

7.5.1 Achieved Design Goals

With GRefExSel, we aim to tackle more complex instructions that require videos for instruction comprehension. Although the proposed model has not yet addressed multi-modal sense-making to potentially include pointing gestures, we have introduced dynamic model optimization via the FrameSkip paradigm to potentially achieve both the low-latency and battery powered sense-making design goals.

It is important to note that the evaluations in this chapter are limited to a server-grade computer, and the proposed models are yet to be tested on a pervasive device. Therefore, we have not yet determined if GRefExSel can achieve the goal of low-latency sense-making within the desired latency of less than 1 second. However, it is worth mentioning that GRefExSel runs 4.32 times faster than STCAT, which may translate to similar gains on a pervasive device in terms of latency and processing energy. Such approximately 4x gains in processing energy are significant, especially in battery-powered scenarios, considering that a typical smart glass or robotic platform can only operate for around 2-3 hours on a single charge.

7.6 Summary

In this chapter, we have showcased the viability of dynamic model optimization for understanding video-based instructions. These models are crucial because not all instruction comprehension scenarios can be effectively addressed by analyzing a single static image; rather, they often necessitate the analysis of a sequence of image frames or videos. Models tailored for this task, such as STVG models, tend to be more resource-intensive due to their requirement to process a continuous stream of images and simultaneously execute spatial and temporal reasoning components. Thus, supporting low latency and low power execution of the traditional STVG models on a pervasive device is a challenging problem. To support this, we investigate a dynamic model optimization technique termed FrameSkip which dynamically chooses a set of anchor frames, conditioned on the language instructions. Complex transformer encoders and decoders are then executed over these selected anchor frames for gains in overall latency. In summary, we make the following key contributions.

- **Adaptive Frame Selector (AFS)**: We propose a lightweight mechanism that detects STVG anchor-frames in a video using a windowed local sub-attention approach for encoding temporal dependencies along with the multi-modal features. By using the identified anchor-frames and the delta features, we demonstrate that we can avoid executing the computationally expensive, transformer based encoder-decoder blocks for nearly 65% of the frames, with negligible impact on the overall accuracy. We call this our key principle of ***FrameSkip***. Using a qualitative analysis, we also identify several key factors resulting in a particular frame being classified as an anchor-frame.

- **Modified loss function to reward frame skipping**: We propose a modification to the standard STVG loss function used in literature to reward the network for skipping video frames. This parameter is tunable to select aggressive or conservative FrameSkip behaviour.

- **GRefExSel model for STVG**: We propose GRefExSel, an efficient STVG model that performs grounding 4.4 - 5x faster (in terms of Frames Per Second) and consumes

20% less memory while maintaining an accuracy that is just 0.55% short of STCAT on the human-centric HC-STVG [97] dataset and 1% lower on the VidSTG dataset [126]. GRefExSel uses the FrameSkip paradigm to execute the encoder-decoder blocks exclusively on selected anchor-frames to drastically reduce the latency and memory requirements. In contrast to the fast(visual)-slow (multi-modal) two-stream approach used by TubeDETR [114], GRefExSel computes multi-modal delta features that are directly fed to the prediction head of the STVG pipeline, completely by-passing the encoder-decoder blocks for non-anchor frames. GRefExsel's inference latency thus grows significantly less rapidly with increasing video length than TubeDETR.

Chapter 8

Exploiting Event Cameras for Efficient Video Grounding

In the preceding chapter, we delved into the challenge of employing STVG models in the context of comprehending video-based instructions. More specifically, we investigated the FrameSkip paradigm, wherein a shallow DNN module known as Adaptive Frame Selector (AFS) was introduced to identify a set of pivotal frames.

However, the evaluation of GRefExSel and GRefExSel-lite models was conducted on a server-grade computer, focusing solely on their impact in reducing latency and memory requirements. Therefore, in this chapter, our primary objective is to enable STVG on a pervasive device while simultaneously reducing latency, processing, and sensing energy overheads.

To achieve this goal, we introduce a STVG system called NeuroViG (**Neuro**morphic **Vi**sual **G**rounding). NeuroViG draws inspiration from the FrameSkip approach, which strategically selects a set of anchor-frames to limit complex processing to these selected frames. The aim is to reduce both latency and processing overhead. A key distinction of this approach compared to GRefExSel is its utilization of an event camera [12], characterized by considerably lower sensing overhead. This event camera serves as a trigger sensor, allowing for the judicious activation of an RGB camera only when necessary. Consequently, this approach drastically

reduces latency, processing, and sensing overhead associated with running STVG models, even on resource-constrained devices.

8.1 Introduction

In chapter 7 we introduced the task of STVG and its applicability to understanding complex human instructions which can only be characterized through a video stream instead of a single static image frame. While GRefExSel and GRefExSel-lite models introduced in the previous chapter significantly reduced the latency and memory requirements of running these complex models, they are still not suitable for continuous execution on a resource-constrained pervasive device. In addition to the processing overheads, which is extensively studied in the chapter 7, traditional STVG models require continuous input from an RGB camera. Such continuous RGB sensing further increases the sensing energy overhead, and may often be unnecessary. Thus, in addition to reducing the processing overhead, it is necessary to develop techniques that can reduce the sensing overhead. To tackle this, we now explore the potential of employing an alternative low-power sensor, namely an event camera, to capture visual context. This approach dynamically triggers computational components in STVG, facilitating low-power and low-latency execution of complex video-based instruction understanding, even on pervasive devices.

More concretely, in this chapter, we propose *NeuroViG* - a system that can support the on-device execution of an STVG task on a pervasive device, with significantly lower processing and sensing overheads. Our key insight is that it is often sufficient to execute the STVG pipeline on a small set of candidate moments/frames within a given video (e.g., when we see a frame that contains relevant objects or gestures mentioned in the accompanying text). Therefore, in this approach, instead of continuously capturing images with the RGB camera, we use an event camera such as the Dynamic Vision Sensor (DVS) [60] to trigger the RGB camera only when required. In contrast to traditional RGB cameras which work at a fixed frame rate, event cameras are associated with a dynamic frame rate dependent on the scene changes in the environ-

ment. They also have several other advantages compared to traditional cameras: a) low latency in the order of microseconds, and b) low power consumption (0.31W in comparison to 1.1W in traditional cameras, based on our empirical evaluations with a Prophesee V2 event camera and Kinect DK RGB camera). Despite these benefits, completely replacing the traditional camera with an event camera is not feasible, since they may not capture static components of a scene as well as a traditional camera. Furthermore, current computer vision pipelines are tailored for sense-making using traditional RGB camera input and not an event-camera stream. Hence, we opt for an opportunistic triggering based mechanism that harnesses the advantages of both an event camera and the RGB-based SOTA models for STVG. In this approach, the RGB camera helps capture static scene components and *key moments* in a video, whereas an event camera provides a low power approximation of the redundant frames of the video as well as more efficient identification of relevant dynamic changes in the video scene.

To facilitate the decision to trigger the RGB camera, we have developed a lightweight early estimator called ***Adaptive Frame Selector (AFS)***. This estimator takes the event-camera stream and the textual query as inputs to predict key moments, known as *anchor-frames*, within the video. While similar to the AFS introduced in chapter 7, which utilized RGB frames and language instructions for identifying anchor-frames, in this instance, we have modified the AFS to employ a event-frame representation [73] of the event-camera stream and the language instruction to identify anchor-frames. Upon detection of these anchor-frames, the RGB camera is triggered.

Furthermore, we propose a lightweight STVG model, called *NeuroViG*, which combines AFS with the transformer encoder-decoder architecture used by state-of-the-art models for STVG. *NeuroViG* significantly reduces processing overheads by executing the complex transformer encoder-decoder pipelines exclusively on the anchor frames predicted by AFS. For the non-anchor frames, AFS provides a feature representation (referred to as delta features) of the visual changes between the current frame and the last identified anchor frame. These delta features can then be added to the encoder output (attention-based features) of the previous anchor frame to generate an approximation of the attention-based features for each non-anchor frame.

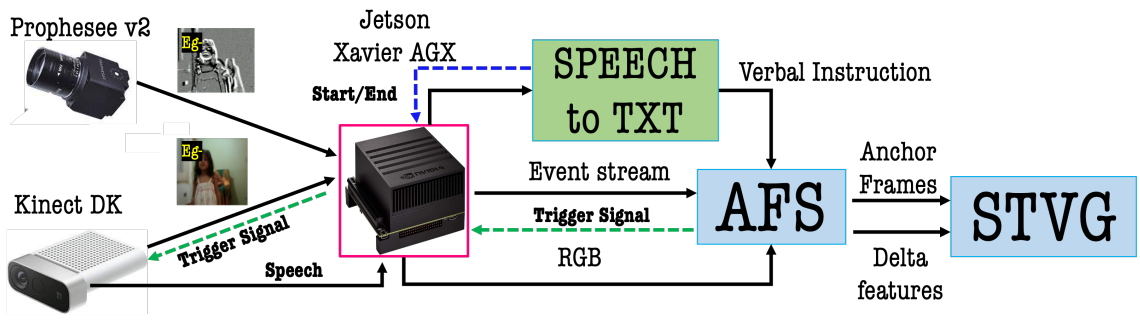


Figure 8.1: *NeuroViG* System Architecture

8.2 *NeuroViG* System

We now describe the *NeuroViG* system proposed in this work. We propose a hybrid sensing system with an event camera and an intermittently triggered RGB camera. Our key idea is to use the RGB sensing stream for some key moments in the video which we termed as *anchor frames* similar to the chapter 7. These anchor frames may represent frames which either will not be captured well through an event camera (static scenes which may generate few events) or the appearance or disappearance of relevant objects or actions. By employing a similar approach of Frame-Skipping as proposed in previous chapter, we further identified that such adaptive sensing can be extended to the processing pipeline. Precisely, we limit the compute-heavy transformer encoder-decoder blocks only on frames associated with such anchor moments. While we could still use the same AFS proposed in chapter 7 to identify these anchor moments with a lower latency, there is still an additional challenge to overcome: the event and traditional RGB cameras have different data representations that must be reconciled within a unified DNN model. Inherently, output from an event camera represents scene changes across frames. Thus, we assume that the processing block from the event camera represents a differential feature representation giving scene changes in respect to the previous frame and the last identified anchor frame. A full blown STVG pipeline is only executed over the identified anchor frames and the rest of the redundant frames are approximated with this computed delta representation. For identifying these anchor frames and delta feature representations, we propose a modified ***Adaptive Frame Selector*** (AFS) described later in detail.

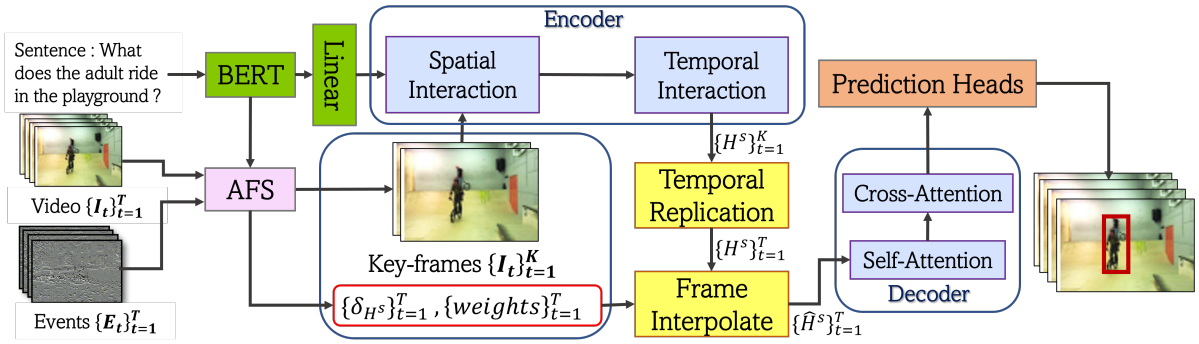


Figure 8.2: Overall architecture of STVG pipeline of *NeuroViG* system

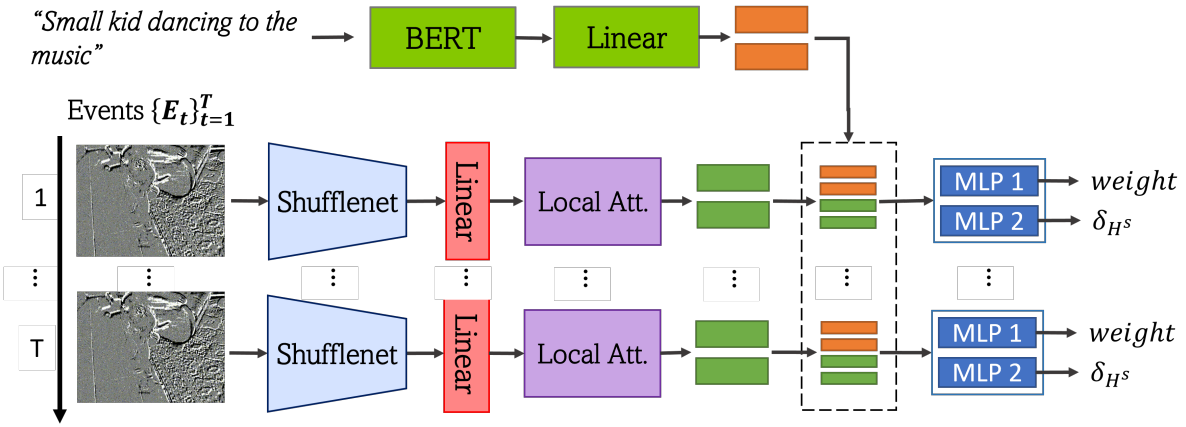


Figure 8.3: Architecture of modified Adaptive Frame Selector

8.2.1 Event-based Adaptive Frame Selector

In Figure 8.3, AFS takes a series of event frames and language features as input. The raw event output of the Prophesee V2 camera is described by a stream of x, y, p, t, where $p \in 0, 1$ describes the polarity of the event in either the negative or positive direction. We convert this stream-based event representation into a frame representation of shape $[1 \times 448 \times 448]$ by aggregating all the events received in a time frame of 33ms, resulting in a fixed 30 FPS input to the AFS model. This frame representation is then passed through a ShuffleNet network for visual encoding, resulting in a 2-dimensional feature vector of size $\mathscr{R}^{T \times D}$, using a fully connected layer with an output dimension of $D = 256$. This differs from the RGB-based AFS, where the RGB frames were passed through the first layer of ResNeT before being reused in the subsequent STVG pipeline. In the event-based AFS, we use a separate ShuffleNet to handle the modality difference between event frames and RGB frames.

These feature vectors are then passed through a window-based local attention network [18] with a window size of 10 to model the temporal dependencies in the local neighborhood, resulting in $V_t \in \mathscr{R}^{T \times D}$. Each visual feature V_t is concatenated with the language features computed using a ROBERTa model. Finally, AFS utilizes two separate inexpensive Multi-Layer Perceptron (MLP) models with 3 layers and dropout for regularization (dropout probability=0.5). One of these MLPs predicts whether a given frame is an anchor-frame or not. The other MLP obtains a feature representation of the visual differences between the current frame and the nearest anchor-frame, referred to as the *delta feature* representation. These delta features are then used to interpolate the encoder output to obtain cross-modal attention-based features for non-anchor frames. These features can be used by the decoder and the prediction heads of the STVG pipeline to predict spatiotemporal tubes. The operation of the two MLPs can be expressed as follows:

$$\begin{aligned} weights = Gumbell(MLP[F_l, E_t^f]) \;\; where \;\; weights \in \mathscr{R}^{T \times 2} \\ \delta_{H^s} = MLP[F_l, E_t^f] \;\; where \;\; \delta_{H^s} \in \mathscr{R}^{T \times D} \end{aligned} \tag{8.1}$$

In the above equation, F_l refers to the language features taken from the same text encoder in STCAT, E_t^f refers to the visual features obtained after computing local attention for an event stream in frame representation of length T frames, while *weights* determine whether the respective frame is an *anchor-frame* or a *redundant-frame*. By following a similar approach as in GRefExSel, we use Gumbell-Softmax activation function, which returns discrete 0 or 1 value for elements in *weights*. When $weights_i[0] = 1$ and $weights_i[1] = 0$, i$^{\text{th}}$ event-frame is deemed to be a *anchor-frame*, activating the RGB camera. When $weights_i[0] = 0$ and $weights_i[1] = 1$, i$^{\text{th}}$ event-frame is deemed to be a *redundant-frame*, keeping the RGB camera off. We also assume that the first frame of the video is always an anchor-frame (Thus, RGB camera is always activated at the start once). The output of the second MLP, denoted by δ_{H^s}, represents the delta features that were described earlier.

8.2.2 Loss Function

We follow the same loss function proposed in 7.2 for rewarding the model for skipping more RGB frames. Similar to GRefExSel, *frame-skip* parameter S determines the percentage of RGB frames that are considered to be redundant in the video. A value closer to 1 would encourage aggressive skipping behaviour, thereby reducing computations and reducing sensing overhead. A value closer to 0 would encourage conservative skipping behaviour resulting higher computational load and sensing overhead.

8.2.3 *NeuroViG* STVG pipeline

We propose an optimized STVG model within the *NeuroViG* system, as depicted in Figure 8.2. The central principle of this model is to activate the traditional camera only for the anchor-frames and send them through an expensive STVG pipeline. Meanwhile, input from an event camera is utilized alongside a late interpolation approach (explained in Equation 7.3) to obtain the corresponding feature representation for the redundant frames.

A key difference of this *NeuroViG* STVG pipeline compared to the GRefExSel model is the additional event input to the pipeline, which is utilized by the event-based AFS for identifying the anchor frames. Furthermore, in GRefExSel, we executed the frame interpolation after the transformer decoder step. However, in this approach, primarily motivated by the modality difference between events and RGB frames, we execute the frame interpolation before the decoder.

In addition to these differences, we maintain the same computations as explained in section 7.3 of chapter 7.

8.2.4 *NeuroViG* Prototype Implementation

Figure 8.1 shows our proposed system. We use Prophesee V2 [12] as the event camera and Kinect DK [1] as our representational RGB camera and microphone. Both these devices are connected to a Jetson Xavier AGX device [6], which executes both AFS and the rest of the STVG pipeline. Both these devices are not supported officially for ARM-based devices. Thus, for Prophesee V2, we use their open-sourced SDK, OpenEB instead of the official MetaVision SDK. For the Kinect DK, we use a cross-compiled version of their SDK for the ARM-based devices. AFS and the rest of the STVG pipeline is implemented using PyTorch. The microphone of the Kinect DK remains continuously active, and we employ a real-time speech-to-text engine [2], running on the AGX device, to convert speech into textual commands. The *NeuroViG* system is initiated upon the user's command, typically with the phrase 'start,' and is subsequently terminated upon receiving the user's 'done' command. This action triggers the activation of the event camera. We assume the initial frame is always designated as an anchor-frame, leading to the activation of the RGB camera for a single frame alongside the event camera. AFS then takes the output from the event camera and textual query and send a trigger signal to the Kinect DK camera once it anticipates an anchor moment in the video. Such intermittent triggering is also associated with a startup delay of 120ms. However, current STVG pipelines are usually executed at 5 FPS, thus we can safely assume that the startup delay of 120ms, is adequate to operate at 5 FPS without missing frames.

8.3 Results

We proceed to experimentally evaluate *NeuroViG* against alternative state-of-the-art STVG models. In general, we adhere to the same evaluation methods utilized in prior work [114, 48]. In addition to the standard metrics, we will also consider latency, frames per second (FPS), sensing energy, processing, and memory consumption as additional performance metrics on a Jetson AGX device.

Similar to chapter 7, we utilize the VidSTG and HC-STVG benchmark datasets in our evaluations. However, all of the existing STVG datasets are traditionally video-based. To obtain the event-based counterpart, we employ V2E simulator [46], which synthesizes realistic event camera data from any real (or synthetic) conventional frame-based video. Consequently, we use V2E to convert the existing videos in VidSTG and HC-STVG to event-based representations.

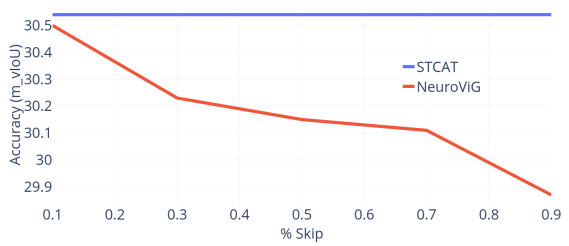
8.3.1 Characterizing *NeuroViG* Performance with Frame-Skip Parameter

During the training process of our proposed AFS and *NeuroViG* , we introduced a reward loss mechanism to incentivize the model to skip traditional RGB frames, aiming to reduce sensing and processing overhead. In this study, we assess the performance of the *NeuroViG* system while varying the frame-skip parameter. Our goal is to optimize the balance between accuracy and energy efficiency.

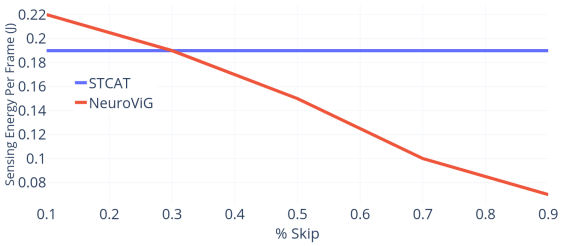
Initial measurements indicated that the Prophesee V2 and Kinect DK cameras draw an average power of 0.31W and 1.125W, respectively. It was evident that utilizing an event camera should ideally reduce sensing power consumption by approximately 4 times.

To evaluate this, we first examined how accuracy of the *NeuroViG* model on the VidSTG dataset fluctuates with different frame-skip parameters, as illustrated in Figure 8.4(a). Subsequently, we assessed how the sensing energy per frame varies with the frame-skip parameter, as shown in Figure 8.4(b). However, it's important to note that sensing energy is just one component contributing to the overall energy overhead of the *NeuroViG* system.

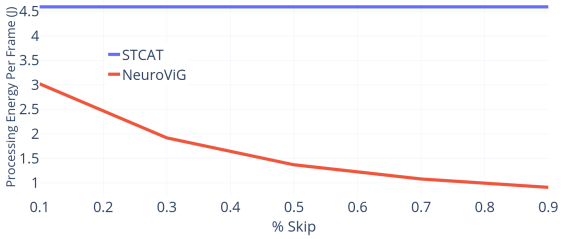
We also evaluated the processing energy overhead of the system for different values of the frame-skip parameter, as presented in Figure 8.4(c). Our findings indicate that increasing the frame-skip parameter reduces both sensing and processing energy overhead but comes at the cost of lower accuracy. Conversely, setting the frame-skip parameter below 0.3 results in higher sensing energy consumption compared to a baseline system using only a traditional RGB cam-



(a) m_vIoU vs frame-skip parameter



(b) Sensing energy per frame vs frame-skip parameter



(c) Processing energy per frame vs frame-skip parameter

Figure 8.4: Variation of evaluation metrics of *NeuroViG* system vs frame-skip parameter

era.

Based on this comprehensive study, we opted to select a frame-skip parameter of 0.7. This choice strikes a balance between energy efficiency and accuracy, aligning with our goal to optimize the energy-accuracy trade-off.

8.3.2 Performance on Benchmark Datasets

Based on our previous study, we then set the frame-skip parameter as 0.7, to achieve a balance between accuracy and energy trade-off. Tables 8.1 and 8.2 then summarize the performance of the proposed *NeuroViG* models against benchmark STVG models on VidSTG and HC-STVG datasets respectively. From our experiments, we observe that the AFS mechanism allows *NeuroViG* to select just about 35% of the video frames as anchor frames, enabling the remaining frames to bypass the encoder-decoder blocks. However, this results in a minor drop in their accuracy when compared to STCAT. The m_vIoU of *NeuroViG* is just 1.08% shy of STCAT on the human-centric HC-STVG dataset. On the more general VidSTG dataset, *NeuroViG* achieves an

Table 8.1: Performance comparisons of the state-of-the-art on the VidSTG test set (%)

Methods	Declarative Sentences					Interrogative Sentences				
	m_tIoU	m_vIoU	vIoU@0.3	vIoU@0.5	% frames skipped	m_tIoU	m_vIoU	vIoU@0.3	vIoU@0.5	% frames skipped
Two-Stage:										
STGVT [97]		21.62	29.80	18.94	0	-	-	-	-	0
One-Stage:										
STVGBert [93]		23.97	30.91	18.39	0		22.51	25.97	15.95	0
TubeDETR [114]	48.10	30.40	42.50	28.20	0	46.90	25.70	35.70	23.20	0
STCAT [48]	50.82	33.14	46.20	32.58	0	49.67	28.22	39.24	26.63	0
GRefExSel (Ours)	49.19	32.36	45.05	31.57	64	49.02	27.27	38.51	25.77	61
NeuroViG (Ours)	**49.01**	**32.15**	**44.89**	**31.06**	**66**	**48.89**	**27.12**	**38.09**	**25.43**	**64**

m_vIoU that is just about 1.1% short of STCAT for the interrogative sentences and about 0.99% short of STCAT for the declarative sentences. When compared to the TubeDETR, *NeuroViG* is able to achieve higher m_vIoU on all the benchmark tasks and also achieve multi-fold improvement in latency and energy (As shown in Table 8.3). Overall, we believe that this drop in accuracy is tolerable in light of the fact that nearly 65% of the frames by-passed the RESNET backbone, transformer encoder and were approximated with an event camera representation instead of a conventional RGB input. We believe that this will translate into a significant savings in latency, energy and memory requirements when dealing with longer untrimmed videos in practical use-cases.

We then measured the average FPS, sensing and processing energy per frame on Jetson AGX and compared them against STCAT and GRefExSel in table 8.3. We found that *NeuroViG* runs 3.82x faster, consumes 2.2x less sensing energy and 4.25x less processing energy than STCAT. Overall, this results in *NeuroViG* being 4.08x energy efficient than the current state-of-the-art STVG approach. Evidently, *NeuroViG* and GRefExSel have comparable accuracy, processing energy and FPS , but *NeuroViG* runs with 2.2x less sensing energy than GRefExSel. This results in a further $\sim$8.5% reduction in overall energy overhead compared to GRefExSel. Although the overall energy gains when comparing GRefExSel and *NeuroViG* may seem marginal, we believe that this 8.5% reduction could lead to a significant increase in usage time in a battery-powered pervasive implementation of STVG. In Figure 8.5, we plot the latency of STCAT against *NeuroViG* system and observe that the latency of each of these models increases steadily with the number of frames. However, the rate of increase of latency for *NeuroViG* is significantly lower than that of RGB-only natural language interface equipped with STCAT model.

Table 8.2: Performance comparisons of the state-of-the-art on the HC-STVG test set (%)

Methods	m_tIoU	m_vIoU	vIoU @0.3	vIoU @0.5	% frames skipped
Two-Stage:					
STVGT	-	18.15	26.81	9.48	0
One-Stage:					
STVGBert	-	20.42	29.37	11.31	0
TubeDETR	43.70	32.40	49.80	23.50	0
STCAT	49.44	35.09	57.67	30.09	0
GRefExSel	48.89	34.23	56.19	29.26	63
NeuroViG	**48.65**	**34.01**	**56.12**	**28.91**	**65**

Table 8.3: Accuracy vs FPS vs Energy trade-off for *NeuroViG* against other baseline STVG systems

Model	**m_vIoU**	**FPS**	**Energy per frame (J)**		
			Sensing	**Processing**	**Total**
STCAT	30.54	2.2	0.22	4.59	4.81
GRefExSel	29.92	8.5	0.22	1.06	1.28
NeuroViG	29.63	8.4	0.10	1.08	1.18

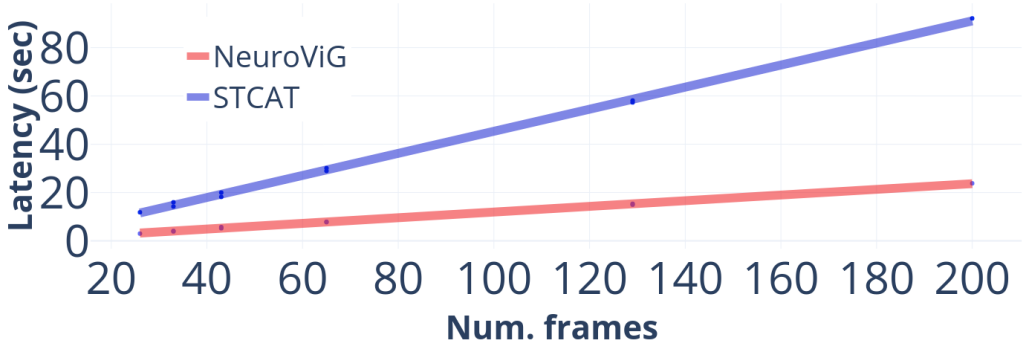


Figure 8.5: Latency vs number of video frames for *NeuroViG* against STCAT

8.3.3 Other variations of *NeuroViG*

We also considered a few additional variations of *NeuroViG* to evaluate the efficacy of our hybrid sensing mechanism.

Random skipping: In this, instead of the *weights* predicting the anchor-frames, we randomly picked 35% of the RGB frames from the video. Rest of the delta features from AFS remain the same. We refer to this in table 8.4 as *NeuroViG*-Random

Event-only skipping: AFS takes both language and event data as input. This variation only feeds event features to the AFS. Idea is to evaluate the performance when the AFS is only capable of seeing scene changes, but not key-words found in language instructions. We refer to this in table 8.4 as *NeuroViG*-Vision

Event-only STVG: Some may question the rationale behind employing a hybrid sensing mechanism that incorporates two sensors (RGB and Event camera) for vision input, especially when we could argue that an Event camera alone might suffice for STVG task.

Based on our observations, such a system would reduce the sensing overhead 4x. Thus, we propose an event-based STVG (which does not utilize RGB frames at all) derived from STCAT. In this variation, we simply replaced the RGB input with frame representation of the event input. We refer to this as Event-STCAT.

Based on the table 8.4, it is evident that the performance of *NeuroViG* is significantly degraded when selecting key-frames at random. This suggests that the careful selection of key-frames is paramount to the performance of *NeuroViG* system. With *NeuroViG*-Vision, we seek to evaluate whether just using visual feature on AFS is enough to identify key-frames. Just using visual features limits the capability of AFS to make an early decision of when the target object appears and only allows to perceive changes in scene. As shown in table 8.4, while the performance degradation is not as dramatic as *NeuroViG*-Random, the drop in accuracy is still substantial compared to *NeuroViG* . This demonstrate the importance of both language and visual features

Table 8.4: Ablation study of *NeuroViG*

Methods	m_tIoU	m_vIoU	vIoU@0.3	vIoU@0.5
NeuroViG	48.62	29.63	41.67	28.07
NeuroViG-Random	31.12	16.15	20.21	14.76
NeuroViG-Vision	40.14	22.98	33.54	21.12
STCAT-Event	34.17	19.76	28.16	17.19

in AFS to predict the key-frames in the video. With Event-STCAT, we seek to evaluate whether the current STVG pipeline can be completely replaced with just using an event camera. While such a system will reduce the sensing overhead by 4x, there is a significant degradation of performance observed with such modification. This suggests that, while event cameras can provide a good energy-aware approximation to traditional RGB cameras when applied to STVG task, RGB/Event hybrid sensing system is paramount for accurate comprehension of STVG instructions and queries.

8.4 Discussion

We introduced NeuroViG as a refinement to GRefExSel, building upon the concepts outlined in the preceding chapter. Our primary objective was to leverage event-camera streams to identify anchor moments, subsequently triggering the activation of the more power-intensive RGB camera. Additionally, we outline the following strategies as potential future directions to enhance the efficiency of STVG.

- **Event-only STVG:** One might argue that we could achieve greater savings in sensing overhead by proposing an STVG model designed to work exclusively with an event stream. In Table 8.4, we demonstrated that merely replacing the STCAT's RGB input with the frame representation of an event stream led to subpar accuracy. However, SNNs are commonly used for processing event streams, and they have shown strong performance in various vision-based tasks. Therefore, it is plausible to consider exploring a dedicated STVG pipeline with SNNs in the future, which may lead to improved system performance.

- **Gesture-enhanced STVG:** Similar to GRefExSel, we anticipate that NeuroViG can readily adapt to support the comprehension of human gestures, an integral aspect of natural human interactions. In such scenarios, event-based AFS can be tailored to capture key frames of human gestures, providing significant data for the STVG pipeline.

8.4.1 Achieved Design Goals

In this chapter, we introduced NeuroViG, a refinement of GRefExSel from the previous chapter. While neither NeuroViG nor GRefExSel has yet addressed the multi-modal sense-making design goal, we specifically evaluated latency and energy performance on a pervasive device for tackling more complex instruction requiring video data.

First, we evaluated the performance of GRefExSel and other STVG baseline models on a pervasive device. The results show that STCAT, GRefExSel, and NeuroViG have latencies of 454.55 ms, 117.65 ms, and 119.05 ms per frame, respectively. Considering that existing work uses videos sampled at 5FPS for STVG, this means that STCAT, GRefExSel, and NeuroViG can handle videos of length 0.44 seconds, 1.70 seconds, and 1.68 seconds, respectively, within the desired 1-second comprehension latency. Thus, even with dynamic model optimizations, the current approach is insufficient for continuous applications because these models utilize transformer-based architectures that require the entire video sequence as input. As a result, actual STVG comprehension can only occur after capturing the whole video sequence. To address this limitation, we plan to extend these models to support long untrimmed videos and streaming videos, allowing comprehension to happen simultaneously as the video data is captured. This extension will further enhance interactivity and help achieve the goal of low-latency sense-making.

Regarding battery-powered sense-making, GRefExSel and NeuroViG have improved total energy consumption by 3.75x and 4.07x, respectively, compared to STCAT. This translates to similar improvements in operating times for executing these models on pervasive devices, given the computationally demanding nature of these models.

8.5 Summary

In the previous chapter, we delved into GRefExSel, a dynamic model optimization approach for Spatio-Temporal Video Grounding (STVG) tasks. While the FrameSkip approach enabled GRefExSel to achieve significant reductions in latency, the evaluations were primarily conducted on a server-grade computer. Therefore, in this chapter, we advance our exploration with *NeuroViG* , an enhanced version of GRefExSel designed to operate efficiently even on pervasive devices.

Our primary strategy involves leveraging an event camera as an alternative sensor for capturing visual context and as a trigger sensor. This enables us to achieve low latency and low energy overheads, even on pervasive devices, facilitating more sophisticated video-based instruction understanding. We introduce further optimizations to the GRefExSel model to propose the *NeuroViG* system, leveraging an event camera to intelligently trigger the more power-hungry traditional RGB camera. To the best of our knowledge, *NeuroViG* represents the first system to harness the capabilities of a neuromorphic event camera for efficient STVG.

In summary, we make the following key contributions.

- ***NeuroViG* – an efficient system for STVG in a pervasive setting**: We believe that *NeuroViG* is the first system to execute the STVG task on a pervasive device (Jetson AGX device), running at 8.2 FPS while consuming a total energy of 1.18J per frame. This is $\sim$ 3.82x faster and 4.08x energy efficient than the state-of-the-art STCAT model that uses only an RGB Camera. Our system thus demonstrates the benefits of using an event camera for opportunistically triggering an a traditional RGB camera to perform complex tasks such as STVG on pervasive devices. Notably, *NeuroViG* maintains an accuracy that is just 0.81% short of STCAT on the human-centric HC-STVG [97] dataset and 0.43% lower on the VidSTG dataset [126].
- **Event camera based Adaptive Frame Selector**: For *NeuroViG*, we have developed a novel Adaptive Frame Selector that combines the event camera data with features from

the natural language instruction to identify ‘anchor-frames’. These anchor-frames denote critical moments within the video that are pivotal for accurately grounding the natural language instruction. Our modified loss function for the AFS enables us to choose between various levels of aggressive or conservative frame-skipping behaviour in the STVG model to exploit the trade-off between accuracy and overall energy consumption (sensing + processing).

Chapter 9

Conclusion and Future Directions

In this chapter, I conclude this dissertation by summarizing the main contributions and outlining some of the possible extended use cases of the proposed technologies and key future directions.

9.1 Summary of Contributions

In this dissertation, I showcase the integration of Referring Expression Comprehension (REC) models to support instruction understanding in Human-AI interactive tasks. To achieve this, we focus on the object acquisition task as a prime example and propose REC models that are *multi-modal*, *real-time*, and *energy efficient*.

A significant portion of the proposed approaches human instruction understanding in this thesis revolves around the concept of dynamic model optimization which is motivated by the fundamental premise that ***natural human instructions are associated with comprehension tasks of varying complexity***. Here, our proposed methods fundamentally first perform quick and inexpensive estimation of the task context/complexity through a shallow DNN to dynamically route the execution pathway, thereby achieving improvements in latency and energy overheads when executing these models, even on pervasive devices.

9.1.1 M2Gestic

In chapter 3, I introduced the M2Gestic system designed to comprehend object acquisition task instructions through a combination of language and pointing gestures. While we were yet to employ any dynamic model optimizations in this chapter, our primary evaluation of M2Gestic aimed to assess the effectiveness of pointing gestures, particularly in ambiguous environments where high object clutter or insufficient verbal instructions could hinder accurate comprehension of the target object. To this end, we made the following key contributions.

- *Developing a Multi-modal Target Selection Algorithm:* We introduce M2Gestic (pronounced 'majestic'), a technique based on knowledge graphs for understanding instructions. M2Gestic incorporates: (a) a neural approach (RNN-based) to generate machine-understandable commands from natural language, (b) a vision-based clustering mechanism to represent spatial relationships, and (c) a fusion mechanism to rank object suitability based on spatial alignment with pointing locations.
- *Demonstrating the Effectiveness of M2Gestic-based Comprehension, with and without Pointing:* Using a standard dataset, we first show that text-only instructions achieve 61.12% accuracy in target selection, compared to 73.64% for humans. Then, through lab and crowd-sourced studies, we illustrate how incorporating pointing gestures enhances comprehension accuracy. Pointing from close distances improves human accuracy to 77.5%, with a slight decrease as distance increases. For AI/robotic agents using M2Gestic, comprehension accuracy improves from 61.12% to 74.75% when the instructor is close to the objects. We also demonstrate how a distance-weighted variant of M2Gestic ensures robustness, maintaining performance even at larger distances.

9.1.2 SoftSkip

SoftSkip introduced in chapter 4 was our first effort on incorporating dynamic model optimizations for language and vision based REC models. To showcase the practicality of dynamic

model optimizations for REC tasks and to enable low latency and low power execution on pervasive devices, we introduced a SoftSkip mechanism. This led to the development of a novel REC model called LGMDP. This approach targets to jointly reduce both latency and processing energy with merely 1% loss in accuracy. Key to this approach is to use language features as a pivot to dynamically choose the relevant computational blocks in the REC pipeline. In summary we made the following key contributions.

- We present LGMDP, a novel run-time DNN optimization approach tailored for multi-modal tasks like REC. LGMDP stands out as the pioneering model employing textual features to streamline computations in visual processing and subsequent multi-modal fusion stages. It introduces SoftSkip, a novel concept preserving features across various visual scales by rapidly approximating computational blocks rather than completely eliminating them.

- LGMDP is implemented alongside several competitive state-of-the-art (SOTA) alternatives. Through evaluation on three benchmark datasets (ReferIt, RefCOCO, and Cops-Ref), LGMDP demonstrates superior accuracy-latency tradeoff compared to alternatives. Despite a mere 0.5% drop (from 65.3% to 64.6%) in comprehension accuracy compared to the non-optimized RealGIN baseline, LGMDP achieves over 33% reduction in processing latency on an NVIDIA Jetson TX2 device. Notably, LGMDP achieves 65.37% accuracy at 220 ms latency, surpassing the best performing SOTA uni-modal pruning alternative (58.31% accuracy at similar latency). Moreover, Section 4.3.2 delves into LGMDP's multi-scale SoftSkip mechanism leveraging appropriate textual cues.

- Additionally, we showcase how LGMDP's SoftSkip-based approach synergizes with standard static pruning for ultra-lightweight, real-time REC execution on the Jetson TX2. While experiencing an 18% accuracy reduction compared to RealGIN baseline, the combined model achieves a remarkable 2.75x reduction in latency and 7x reduction in memory overhead. This outperforms static pruning alone, which incurs a 17% accuracy loss with 2.2x and 7x reduction in latency and memory, respectively.

9.1.3 COSM2IC

Following the motivation of SoftSkip, our aim in chapter 5 was to integrate dynamic model optimizations into language+vision+pointing gesture-based REC models. To achieve this goal, we introduced the COSM2IC model, which intelligently switches among multiple available models, each tailored to different levels of complexity across various modalities. We made the following key contributions in this work.

- *Multi-modal ego-centric instruction dataset:* Our curated dataset comprises a vast collection of multi-modal referring instructions, focusing on tabletop scenarios from a first-person perspective captured through a head-mounted camera. We meticulously arrange tabletop objects to introduce varying levels of ambiguity and multi-modal nuances. Notably, alongside other inputs, we include depth images primarily to capture pointing gestures.

- *Improved and diverse multi-modal comprehension models:* In support of COSM2IC's dynamic model-switching paradigm, we develop a range of multi-modal models incorporating verbal, visual, and gestural elements with varying complexities. Enhancing the RealGIN model, we replace its visual backbone with a ShuffleNet-based pipeline and extend attention modules to accommodate depth-image-based gesture input. The resulting RealG(2)IN-Lite model exhibits approximately 2x improvement in processing latency and consumes 7x less memory, with only a marginal 3% accuracy loss. To address comprehension in resource-constrained scenarios, we further devise two lightweight variants of RealGIN-based models.

- *COSM2IC paradigm for efficient multi-modal REC:* We introduce COSM2IC, a novel model-switching paradigm designed to slash instruction processing latency by threefold compared to RealGIN, while maintaining a target object identification accuracy of 76.13%, with only a slight 5.53% deviation from RealGIN. This efficiency is achieved through a lightweight neural network-based model for Task Complexity Optimization Procedure (TCOP), utilizing a blend of visual and language embedding features to clas-

sify task complexity levels and select an appropriate multi-modal inference model for execution. Notably, COSM2IC surpasses previous complexity-reduction methods, offering approximately 12-16% higher comprehension accuracy under equivalent computational complexity.

9.1.4 CAS

While COSM2IC demonstrated remarkable improvements in processing energy overheads and latency, making it directly applicable to interactive human-AI collaborative tasks, it required a LiDAR camera to capture pointing gestures. However, this additional modality significantly increased sensing energy overheads. To address this limitation, in chapter 6, we introduced a novel sensor triggering technique coupled with a dynamic model optimization, termed CAS for multi-modal REC. Key to this approach is to utilize depth camera exclusively as an alternative sensor for capturing visual context. It is triggered only when necessary through the dynamic execution pipeline proposed in this approach. We made the following key contributions.

- *CAS-Based, Multi-Branch Multi-modal Instruction Comprehension Model:* Introducing the CAS paradigm, our DNN-based model comprises multiple branches, each with independent heads of varying complexity, selectively executable. We propose an ”efficiency” metric, balancing accuracy and energy tradeoffs, to identify a branch for early task context estimation and subsequent branch switching. Demonstrating CAS with RealGIN-MH, a modified version of RealG(2)In-Lite, we initially commit to a branch using RGB camera data alone. The integrated complexity detector triggers the energy-intensive depth camera only when necessary, achieving energy savings of approximately 12 times compared to COSM2IC.
- *Reconstruction of Depth Image Keyframe:* Despite significant energy savings from triggered depth sensor activation, the startup-delay often leads to a 3-4% loss in comprehension accuracy. For this, we employ a lightweight neural network to regenerate the user’s

unobserved pointing gesture, utilizing future depth frames. We show a median error of only 36 pixels, effectively resolving the true pointing location.

- *Demonstration of Performance Benefits:* Implementing RealGIN-MH on a Nvidia Jetson TX2 platform, equipped with an RGB camera, microphone, and Intel RealSense L515 LIDAR depth camera, we replay sensor data from the COSM2IC dataset. RealGIN-MH achieves a significant reduction in energy overheads, approximately 12.6 times lower, while maintaining comparable accuracy compared to both RealG(2)In-Lite and COSM2IC.

9.1.5 GRefExSel

We then shifted our focus to understanding video-based instructions which represent a more intricate subset of human instructions not fully graspable through a single static image. Addressing this task necessitates the utilization of Spatio-Temporal Video Grounding (STVG) models. However, these models pose significant resource demands, particularly when deployed on pervasive devices. They require processing a sequence of image frames and executing spatial and temporal reasoning components within a single execution pipeline. Therefore, we contended that the proposed dynamic model optimizations are crucial for enabling these models to operate with low latency and minimal energy overheads. In GRefExSel, as proposed in Chapter 7, our primary approach to dynamic model optimization is a concept termed FrameSkip. Here, we identify a set of key moments in the video, termed anchor frames, which are predicted through a shallow DNN module called the Adaptive Frame Selector (AFS). We made the following key contributions in this work.

- *Adaptive Frame Selector (AFS):* We introduce a lightweight mechanism, AFS, which employs a windowed local sub-attention approach to detect STVG anchor-frames in a video. By leveraging multi-modal features and delta features, we demonstrate the ability to bypass computationally expensive transformer-based encoder-decoder blocks for approximately 65% of frames, with minimal impact on overall accuracy. This principle, termed *FrameSkip*, optimizes processing latency without compromising accuracy. Addi-

tionally, through qualitative analysis, we identify key factors influencing the classification of a frame as an anchor-frame.

- *Modified Loss Function to Reward Frame Skipping:* We propose a modification to the standard STVG loss function to incentivize the network to skip video frames. This adjustable parameter allows for fine-tuning the FrameSkip behavior, enabling either aggressive or conservative frame skipping strategies.

- *GRefExSel Model for STVG:* Presenting GRefExSel, an efficient STVG model, achieving approximately 4.4 - 5 times faster processing speed (in Frames Per Second) and consuming roughly 20% less memory compared to STCAT on the HC-STVG and VidSTG datasets. Despite a slight accuracy reduction of only 0.55% on HC-STVG and $\sim$1% on VidSTG compared to STCAT, GRefExSel incorporates the FrameSkip paradigm to execute encoder-decoder blocks exclusively on selected anchor-frames. This strategy significantly reduces latency and memory requirements. Unlike the two-stream approach of TubeDETR, which exhibits fast(visual)-slow(multi-modal) processing, GRefExSel directly computes multi-modal delta features for non-anchor frames, bypassing the encoder-decoder blocks altogether. As a result, GRefExSel's inference latency scales more favourably with increasing video length compared to STCAT and TubeDETR.

9.1.6 *NeuroViG*

In chapter 8, we introduce enhancements to GRefExSel by integrating a neuromorphic event camera, aiming to jointly reduce latency, processing, and sensing energy overheads. Our specific approach involved leveraging a low-power event camera as an alternative sensor for capturing visual context. We dynamically triggered the RGB camera only when necessary and utilized features computed from the event camera to interpolate features for frames where the RGB camera remained inactive. This optimized Spatio-Temporal Video Grounding (STVG) model, termed *NeuroViG* , represents the initial endeavour to enable STVG on a pervasive device and also the first proposal of an RGB+Event-based model for STVG tasks. We make the

following key contributions with *NeuroViG* system.

- NeuroViG - *An Efficient System for STVG in a Pervasive Setting: NeuroViG* marks a significant advancement as the first system to execute the STVG task on a pervasive device, specifically the Jetson AGX, achieving a processing rate of 8.2 FPS with a total energy consumption of 1.18J per frame. Compared to the state-of-the-art STCAT model utilizing only an RGB camera, *NeuroViG* demonstrates approximately 3.82 times faster processing speed and 4.08 times greater energy efficiency. This underscores the advantages of leveraging an event camera to opportunistically trigger a traditional RGB camera for complex tasks like STVG on pervasive devices. Notably, *NeuroViG* maintains a high accuracy level, trailing STCAT by just 0.81% on the HC-STVG dataset and 0.43% on the VidSTG dataset.

- *Event Camera-based Adaptive Frame Selector:* For *NeuroViG* , we introduce a groundbreaking Adaptive Frame Selector that integrates event camera data with natural language instruction features to identify ”anchor-frames.” These frames represent crucial moments within the video essential for accurately grounding the natural language instruction. Our modified loss function for AFS enables flexible adjustment between aggressive and conservative frame-skipping behaviors in the STVG model, effectively managing the trade-off between accuracy and overall energy consumption (sensing + processing).

9.2 Future Directions

In this dissertation, we propose a range of solutions aimed at facilitating multi-modal sense-making in Human-AI interaction tasks. The majority of the solutions outlined here center around REC models optimized through dynamic model optimizations. Therefore, our primary principle for facilitating optimized sense-making is to embrace dynamic model optimizations. With a similar rationale, I will now outline some future directions and early explorations in this domain.

9.2.1 Dynamic model optimizations on Vision Language Models (VLM)

Since the release of GPT3.5 and GPT4 by OpenAI [5], a plethora of open-source implementations of VLMs [64, 102, 132] has surfaced. These models have demonstrated superior performance, sparking discussion on their potential applicability to understanding object acquisition task instructions. However, it's essential to note that VLMs are built upon transformers, which inherently demand significant computational resources. In our exploration, detailed in chapters 7 and 8, we investigate the application of dynamic model optimization techniques on transformer-based models, introducing GRefExSel and NeuroViG. While we confirm the feasibility of dynamic model optimization in transformer-based models, the pervasive deployment of such models remains challenging due to their larger parameter size. Table 9.1 illustrates the instruction comprehension accuracy of CogVLM [102], a prominent open-source VLM, on the COSM2IC dataset relative to the number of parameters. It's important to note that we utilized a pre-trained CogVLM model for testing on the COSM2IC dataset, while the COSM2IC model itself has been specifically trained on the COSM2IC dataset. Consequently, CogVLM exhibits a relatively subpar performance of 47.2% compared to COSM2IC's 76.1%. Despite the potential benefits of fine-tuning CogVLM on the COSM2IC dataset to improve accuracy, deploying this model on pervasive devices poses a significant challenge due to its 345x higher parameter count (17.64 billion parameters in CogVLM compared to just 51.26 million parameters in COSM2IC) compared to off-the-shelf optimized REC models like COSM2IC.

Current efforts to deploy VLMs on pervasive devices primarily rely on model quantization [31] techniques. This involves reducing the number of bits used to store each weight parameter, typically from 32 bits to 16, 8, or even 4 bits, thereby mitigating memory overhead to accommodate resource-constrained devices. Several VLMs have been statically optimized for execution on mobile devices [30, 11], albeit often at the expense of reduced accuracy. As part of our preliminary investigation, we evaluated the feasibility of deploying a quantized Llava model [64] on a Jetson AGX NX device. Table 9.2 presents a timing analysis of this quantized Llava model on the Jetson AGX NX device. From the analysis, it's evident that generating a single token, even with a quantized VLM, requires a total time of 2.95 seconds. If this VLM is employed

Table 9.1: Accuracy vs number of parameters of CogVLM and COSM2IC

Model	**Accuracy (%)**	**No. of Params**
COSM2IC	76.1	51.26M
CogVLM	47.2	17.64B

Table 9.2: Timing analysis of a quantized Llava model on a Jetson NX

Module in Llava	**Avg. latency (sec)**
Vision + language encoder	1.74
Decoder (Latency reported per generated token)	1.21

for REC tasks, which involve generating 4 tokens for bounding box coordinates, the average time taken would be $1.74 + 4 * 1.21 = 6.58$ seconds. In contrast, COSM2IC accomplishes the same task within 105msec yielding a $\sim$63x faster inference. In Human-AI collaborative tasks, where interactivity and low latency are crucial, such prolonged inference times are far from ideal. Thus, while static model optimization or quantization may offer a viable approach for mobile deployment in future, an alternative research direction could involve exploring dynamic model optimization, akin to the methodologies proposed in this dissertation, to further enhance the performance of these VLMs on such devices.

9.2.2 Supporting long, untrimmed streaming videos for STVG

In our endeavor to enhance video-based instruction understanding, we proposed optimizing existing STVG models with dynamic model optimizations. While our proposed techniques, GRefExSel and *NeuroViG* , showcased in chapters 7 and 8, notably reduced the inference latency per video frame, they, along with existing STVG models, require the entire video as input. In cases where video lengths are relatively short, such as in benchmark datasets like VidSTG and HC-STVG, which present trimmed videos of approximately 40 seconds, this requirement poses no significant challenge. However, in real-world scenarios, video instructions often tend to be lengthy, untrimmed and often acquired continuously from a camera. Consequently, loading such long videos into the GPU memory of a pervasive device to run a transformer-based model would inevitably lead to memory overflow issues. Table 9.3 illustrates the memory over-

Table 9.3: Maximum number of frames supported by state-of-the-art STVG models on Jetson AGX without memory overflow

Model	**Max. frames**
STCAT	260
GRefExSel	390

flow issue, demonstrating that the Jetson AGX device encounters overflow beyond 260 frames for STCAT. Although GRefExSel extends this limit to 390 frames, we may encounter much longer videos in real-world applications. Moreover, considering that the video is continuously acquired from a source, the STVG task necessitates execution without knowledge of future frames and in a continuous streaming fashion.

To address this challenge, Gan et al. [36] introduced the concept of Temporal Video Grounding in Streaming Videos (TSGSV), pioneering a streaming grounding model. Central to their approach is a TwinNet architecture for grounding without future frames, coupled with a language query conditioned feature compressor that continuously compresses historical frames. However, their focus has been primarily on temporal grounding, leaving Spatio-Temporal Grounding in Streaming Videos (STGSV) in a single-stage model architecture as an open challenge. Additionally, while they have demonstrated that their proposed model can achieve grounding without future frames, the videos in benchmark datasets used are still significantly shorter than (less than 5mins) those anticipated in real-world scenarios.

As a potential future direction to address these limitations, we plan to investigate a windowed transformer model designed for Spatio-Temporal Video Grounding (STVG). This model would take a buffer of incoming video frames as input at each time step. However, implementing such a windowed inferencing approach would necessitate a memory element to store anchor moments in the streaming video, conditioned on the language query. In this context, we could leverage our proposed AFS mechanism to identify anchor moments in the video and generate the memory representation based on these frames. This approach could enable more efficient and accurate processing of streaming videos for STVG tasks.

9.2.3 Accommodating additional modalities like gaze and gestures in STVG

In our exploration of natural human interaction, we acknowledge that communication extends beyond language and visual cues to encompass modalities such as gaze and hand gestures, as discussed in Section 1.2. While our study has delved into incorporating pointing gestures into REC tasks in chapters 3, 5, and 6, these have primarily focused on static pointing gestures captured through a single key-frame image. Additionally, we operated under the assumption that users only point at a single target object. Consequently, these simpler static pointing gestures align well with the capabilities of image-based REC models analyzed throughout this dissertation.

However, considering more complex pointing gestures as an example, instructions may not span the entire video but rather be temporally limited. Moreover, users may point at multiple objects during the instruction, necessitating adaptations to STVG models to comprehend such gestures. In this context, anchor frames could refer to frames capturing the user steadily pointing at objects, offering valuable information for visual grounding. Our AFS module could leverage this additional information to classify these frames as anchor frames, enhancing performance.

Beyond pointing gestures, dynamic hand gestures present another facet of human interaction that cannot be adequately captured through a single static image, underscoring the importance of STVG in accommodating such gestures. Regarding gaze, users typically follow the target object with their gaze while issuing instructions. Thus, minor variations in gaze patterns could be integrated as an additional delta feature, akin to the approach proposed in the GRefExSel method.

Bibliography

[1] Azure kinect dk. https://azure.microsoft.com/en-us/products/kinect-dk. Accessed: 2023-10-10.

[2] Cheetah streaming speech-to-text. https://picovoice.ai/platform/cheetah/, . Accessed: 2022-04-11.

[3] High voltage power monitor n. https://www.msoon.com/hvpm-product-documentation. Accessed: 2022-10-08.

[4] Intel realsense technology. https://www.intel.sg/content/www/xa/en/architecture-and-technology/realsense-overview.html. Accessed: 2022-07-20.

[5] Introducing chatgpt. https://openai.com/blog/chatgpt. Accessed: 2024-02-14.

[6] Jetson agx xavier series. https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-agx-xavier/. Accessed: 2023-10-10.

[7] Jetson tx2 module. https://developer.nvidia.com/embedded/jetson-tx2. Accessed: 2022-04-11.

[8] Jetson tx2 module. https://developer.nvidia.com/embedded/jetson-tx2. Accessed: 2021-01-16.

[9] Microsoft hololens 2. https://www.microsoft.com/en-us/hololens. Accessed: 2022-07-20.

[10] Nvidia jetson platforms. https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/. Accessed: 2022-05-16.

[11] Pixel 8 pro – the first smartphone with ai built in – is now running gemini nano. https://store.google.com/intl/en/ideas/articles/pixel-feature-drop-december-2023/. Accessed: 2024-02-15.

[12] Prophesee evaluation kit 2 – hd. https://www.prophesee.ai/event-based-evk-2/. Accessed: 2023-10-10.

[13] Pruning tutorial. https://pytorch.org/tutorials/intermediate/pruning_tutorial.html. Accessed: 2021-01-16.

[14] Hyemin Ahn, Sungjoon Choi, Nuri Kim, Geonho Cha, and Songhwai Oh. Interactive text2pickup networks for natural language-based human–robot collaboration. *IEEE Robotics and Automation Letters*, 3(4):3308–3315, 2018.

[15] Amazon. Mechanical turk. `www.mturk.com`. Accessed: 2019-09-30.

[16] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pages 5803–5812, 2017.

[17] Kittipat Apicharttrisorn, Xukan Ran, Jiasi Chen, Srikanth V Krishnamurthy, and Amit K Roy-Chowdhury. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 96–109, 2019.

[18] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[19] Sourav Bhattacharya and Nicholas D Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 176–189, 2016.

[20] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. *arXiv preprint arXiv:1702.07811*, 2017.

[21] Bastien Boutonnet and Gary Lupyan. Words jump-start vision: A label advantage in object recognition. *Journal of Neuroscience*, 35(25):9329–9335, 2015.

[22] Cynthia Breazeal, Cory D Kidd, Andrea Lockerd Thomaz, Guy Hoffman, and Matt Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *2005 IEEE/RSJ international conference on intelligent robots and systems*, pages 708–713. IEEE, 2005.

[23] Qingqing Cao, Noah Weber, Niranjan Balasubramanian, and Aruna Balasubramanian. Deqa: On-device question answering. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 27–40, 2019.

[24] Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. Temporally grounding natural sentence in video. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 162–171, 2018.

[25] Jingyuan Chen, Lin Ma, Xinpeng Chen, Zequn Jie, and Jiebo Luo. Localizing natural language in videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8175–8182, 2019.

[26] Yi-Wen Chen, Yi-Hsuan Tsai, and Ming-Hsuan Yang. End-to-end multi-modal video temporal grounding. *Advances in Neural Information Processing Systems*, 34:28442–28453, 2021.

[27] Yixin Chen, Qing Li, Deqian Kong, Yik Lun Kei, Song-Chun Zhu, Tao Gao, Yixin Zhu, and Siyuan Huang. Yourefit: Embodied reference understanding with language and gesture. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1385–1395, 2021.

[28] Zhenfang Chen, Lin Ma, Wenhan Luo, and Kwan-Yee Kenneth Wong. Weakly-supervised spatio-temporally grounding natural sentence in video. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1884–1894, Florence, Italy, July 2019. Association for Computational Linguistics.

[29] Zhenfang Chen, Peng Wang, Lin Ma, Kwan-Yee K Wong, and Qi Wu. Cops-ref: A new dataset and task on compositional referring expression comprehension. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10086–10095, 2020.

[30] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023.

[31] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or- 1. arxiv 2016. *arXiv preprint arXiv:1602.02830*.

[32] Jiajun Deng, Zhengyuan Yang, Tianlang Chen, Wengang Zhou, and Houqiang Li. Transvg: End-to-end visual grounding with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1769–1779, 2021.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] Fethiye Irmak Dogan and Iolanda Leite. Using depth for improving referring expression comprehension in real-world environments. *arXiv preprint arXiv:2107.04658*, 2021.

[35] J. M. Foley and Richard Held. Visually directed pointing as a function of target distance, direction, and available cues. *Perception & Psychophysics*, pages 263–268, May 1972.

[36] Tian Gan, Xiao Wang, Yan Sun, Jianlong Wu, Qingpei Guo, and Liqiang Nie. Temporal sentence grounding in streaming videos. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4637–4646, 2023.

[37] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275, 2017.

[38] Dawud Gordon, Jurgen Czerny, Takashi Miyaki, and Michael Beigl. Energy-efficient activity recognition using prediction. In *2012 16th international symposium on wearable computers*, pages 29–36. IEEE, 2012.

[39] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pages 799–804. Springer, 2005.

[40] Boris Gromov, Gabriele Abbate, Luca M Gambardella, and Alessandro Giusti. Proximity human-robot interaction using pointing gestures and a wrist-mounted imu. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8084–8091. IEEE, 2019.

[41] Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International workshop ontoImage*, volume 2, 2006.

[42] Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8393–8400, 2019.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] HTC. Vive. `www.vive.com`. Accessed: 2019-09-30.

[45] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4555–4564, 2016.

[46] Y Hu, S C Liu, and T Delbruck. v2e: From video frames to realistic DVS events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2021.

[47] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[48] Yang Jin, Zehuan Yuan, Yadong Mu, et al. Embracing consistency: A one-stage approach for spatio-temporal video grounding. *Advances in Neural Information Processing Systems*, 35:29192–29204, 2022.

[49] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017.

[50] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[51] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Youngki Lee, Souneil Park, and Junehwa Song. A scalable and energy-efficient context monitoring framework for mobile personal sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):686–702, 2009.

[52] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[53] Casey Kennington and David Schlangen. A simple generative model of incremental reference resolution for situated dialogue. *Computer Speech & Language*, 41:43–67, 2017.

[54] Donnie H Kim, Younghun Kim, Deborah Estrin, and Mani B Srivastava. Sensloc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 43–56, 2010.

[55] Peter Kok, Michel F Failing, and Floris P de Lange. Prior expectations evoke stimulus templates in the primary visual cortex. *Journal of cognitive neuroscience*, 26(7):1546–1554, 2014.

[56] Nikhil Krishnaswamy and James Pustejovsky. Generating a novel dataset of multimodal referring expressions. In *Proceedings of the 13th International Conference on Computational Semantics-Short Papers*, pages 44–51, 2019.

[57] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[58] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344, 2020.

[59] Shen Li, Rosario Scalise, Henny Admoni, Stephanie Rosenthal, and Siddhartha S Srinivasa. Spatial references and perspective in natural language instructions for collaborative manipulation. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 44–51. IEEE, 2016.

[60] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers*, pages 2060–2069. IEEE, 2006.

[61] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in neural information processing systems*, pages 2181–2191, 2017.

[62] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[63] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[64] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

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

[66] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[67] Konrad Lorincz, Bor-rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal Patel, Paolo Bonato, Matt Welsh, et al. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *SenSys*, volume 9, pages 183–196, 2009.

[68] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th International Conference on Embedded Networked Sensor Systems, SenSys 2010*, pages 71–84. ACM, 2010.

[69] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[70] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[71] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[72] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20, 2016.

[73] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5419–5427, 2018.

[74] Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[75] Sven Mayer, Valentin Schwind, Robin Schweigert, and Niels Henze. The effect of offset correction and cursor on mid-air pointing in real and virtual environments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[76] Varun K Nagaraja, Vlad I Morariu, and Larry S Davis. Modeling context between objects for referring expression understanding. In *European Conference on Computer Vision*, pages 792–807. Springer, 2016.

[77] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[78] Suman Nath. Ace: Exploiting correlation for energy-efficient and continuous context sensing. In *MobiSys'12: The 10th International Conference on Mobile Systems, Applications and Services (Best Paper Award)*. ACM, June 2012.

[79] Diederick C Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205, 2017.

[80] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 260–273, 2011.

[81] Rohan Paul, Jacob Arkin, Derya Aksaray, Nicholas Roy, and Thomas M. Howard. Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms. *International Journal of Robotics Research*, 37(10):1269–1299, 9 2018.

[82] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3857–3866, 2019.

[83] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 779–788, 2016.

[84] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[85] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 817–834. Springer, 2016.

[86] Arka Sadhu, Kan Chen, and Ram Nevatia. Zero-shot grounding of objects from natural language queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4694–4703, 2019.

[87] Rosario Scalise, Shen Li, Henny Admoni, Stephanie Rosenthal, and Siddhartha S Srinivasa. Natural language instructions for human–robot collaborative manipulation. *The International Journal of Robotics Research*, 37(6):558–565, 2018.

[88] Sougata Sen, Vigneshwaran Subbaraju, Archan Misra, Rajesh Balan, and Youngki Lee. Annapurna: An automated smartwatch-based eating detection and food journaling system. *Pervasive and Mobile Computing*, 68:101259, 2020.

[89] Shreyas S Shivakumar, Neil Rodrigues, Alex Zhou, Ian D Miller, Vijay Kumar, and Camillo J Taylor. Pst900: Rgb-thermal calibration, dataset and segmentation network. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 9441–9447. IEEE, 2020.

[90] Mohit Shridhar and David Hsu. Interactive visual grounding of referring expressions for human-robot interaction. In *Proceedings of Robotics: Science and Systems*, 2018.

[91] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1731–1740, 2018.

[92] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738, 2013.

[93] Rui Su, Qian Yu, and Dong Xu. Stvgbert: A visual-linguistic transformer based framework for spatio-temporal video grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1533–1542, 2021.

[94] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[95] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7464–7473, 2019.

[96] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[97] Zongheng Tang, Yue Liao, Si Liu, Guanbin Li, Xiaojie Jin, Hongxu Jiang, Qian Yu, and Dong Xu. Human-centric spatio-temporal video grounding with visual transformers. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[98] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[99] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[101] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2320–2329, 2020.

[102] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[103] Weining Wang, Yan Huang, and Liang Wang. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 334–343, 2019.

[104] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[105] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 179–192, 2009.

[106] Dulanga Weerakoon, Vigneshwaran Subbaraju, Nipuni Karumpulli, Tuan Tran, Qianli Xu, U-Xuan Tan, Joo Hwee Lim, and Archan Misra. Gesture enhanced comprehension of ambiguous human-to-robot instructions. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, pages 251–259, 2020.

[107] Dulanga Weerakoon, Vigneshwaran Subbaraju, Tuan Tran, and Archan Misra. Cosm2ic: Optimizing real-time multi-modal instruction comprehension. IEEE, 2022.

[108] Dulanga Weerakoon, Vigneshwaran Subbaraju, Tuan Tran, and Archan Misra. Softskip: Empowering multi-modal dynamic pruning for single-stage referring comprehension. In *ACM Multimedia*, 2022.

[109] David Whitney, Miles Eldon, John Oberlin, and Stefanie Tellex. Interpreting multimodal referring expressions in real time. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3331–3338. IEEE, 2016.

[110] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019.

[111] Shaoning Xiao, Long Chen, Songyang Zhang, Wei Ji, Jian Shao, Lu Ye, and Jun Xiao. Boundary proposal network for two-stage natural language video localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2986–2994, 2021.

[112] Huijuan Xu, Kun He, Bryan A Plummer, Leonid Sigal, Stan Sclaroff, and Kate Saenko. Multilevel language and vision integration for text-to-clip retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9062–9069, 2019.

[113] Masataka Yamaguchi, Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Spatio-temporal person retrieval via natural language queries. In *Proceedings of the IEEE international conference on computer vision*, pages 1453–1462, 2017.

[114] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Tubedetr: Spatio-temporal video grounding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16442–16453, 2022.

[115] Sibei Yang, Guanbin Li, and Yizhou Yu. Dynamic graph attention for referring expression comprehension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4644–4653, 2019.

[116] Zhengyuan Yang, Tianlang Chen, Liwei Wang, and Jiebo Luo. Improving one-stage visual grounding by recursive sub-query construction. In *ECCV*, 2020.

[117] Zhengyuan Yang, Boqing Gong, Liwei Wang, Wenbing Huang, Dong Yu, and Jiebo Luo. A fast and accurate one-stage approach to visual grounding. In *ICCV*, 2019.

[118] Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher. Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 278–291, 2018.

[119] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–14, 2017.

[120] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1031–1042, 2018.

[121] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018.

[122] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016.

[123] Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1247–1257, 2019.

[124] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[125] Zhu Zhang, Zhou Zhao, Zhijie Lin, Baoxing Huai, and Nicholas Jing Yuan. Object-aware multi-branch relation networks for spatio-temporal video grounding. *arXiv preprint arXiv:2008.06941*, 2020.

[126] Zhu Zhang, Zhou Zhao, Yang Zhao, Qi Wang, Huasheng Liu, and Lianli Gao. Where does it exist: Spatio-temporal video grounding for multi-form sentences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10668–10677, 2020.

[127] Yang Zhao, Zhou Zhao, Zhu Zhang, and Zhijie Lin. Cascaded prediction network via segment tree for temporal video grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4197–4206, 2021.

[128] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12993–13000, 2020.

[129] Yiyi Zhou, Rongrong Ji, Gen Luo, Xiaoshuai Sun, Jinsong Su, Xinghao Ding, Chia-Wen Lin, and Qi Tian. A real-time global inference network for one-stage referring expression comprehension. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[130] Zhuyun Zhou, Zongwei Wu, Rémi Boutteau, Fan Yang, Cédric Demonceaux, and Dominique Ginhac. Rgb-event fusion for moving object detection in autonomous driving. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7808–7815. IEEE, 2023.

[131] Alex Zihao Zhu and Liangzhe Yuan. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems*, 2018.

[132] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.