LANGUAGE GUIDED LOCALIZATION AND NAVIGATION

A Dissertation
Presented to
The Academic Faculty

By

Meera Hahn

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science in the
School of Interactive Computing

Georgia Institute of Technology

August 2022

LANGUAGE GUIDED LOCALIZATION AND NAVIGATION

Thesis committee:

Dr. James M. Rehg
College of Computing
Georgia Institute of Technology

Dr. Abhinav Gupta
The Robotics Institute
Carnegie Mellon University

Dr. Dhruv Batra
College of Computing
Georgia Institute of Technology

Dr. Peter Anderson
Research Scientist
Google Research

Dr. Diyi Yang
College of Computing
Georgia Institute of Technology

ACKNOWLEDGMENTS

There are many people who supported me and encouraged me during my PhD. I first would like to thank my family. Without parents, Pranhitha and Steve, I would not be the person I am today. They helped instill in me a strong work ethic and passion for exploration. To my brother Nathan. And to my Uncle Bhagi and Aunt Poorvi who encouraged me to take my first computer science class.

I would also like to thank my undergraduate mentors Dr. Jinho Choi and Dr. Afshin Deghan who introduced me to computer science research, and the fields of computer vision and natural language processing. They also encouraged and supported me during my application to PhD programs and continue to be a source of support and guidance.

At Georgia Tech I have had wonderful mentors and peers who have taught me how to be a successful and impactful researcher in my field. First I would like to thank Dr. James M. Rehg who was my advisor and guided me through this program. I would also like to thank the rest of my committee members, Dr. Abhinav Gupta, Dr. Peter Anderson, Dr. Dhruv Batra and Dr. Diyi Yang, for their fruitful advice and comments. Devi, Dhruv, Peter and Dr. Stefan Lee have all been amazing mentors during my time at Georgia Tech and feel lucky to have gotten the opportunity to learn from them. I have also had the opportunity to collaborate with many people outside of Georgia Tech during my PhD. My internship mentors, Dr. Devendra Chaplot at FAIR, Dr. Asim Kadav at NEC laboratories, and Dr. James Hillis at Facebook Reality Labs as well as all my other collaborators who I did not name, it was a pleasure working with you all and getting to learn from you.

My peers at Georgia Tech have been so inspiring and supportive. The environment they foster is one of collaboration and encouragement. Some of the students I have been lucky enough to work along side of are: Amit Raj, Erik Wijmans, Jonathan Balloch, Sasha Lambert, Sarah Wiegraffe, Vincent Cartillier, Steven Hickson, Andrew Silva, Cole DeLude, Abishek Das, Arjun Majumdar, Duri Long, Samyak Datta, Stefan Stojanov, Ahn Thai, and

Nataniel Ruiz.

Also to all my amazing friends outside of the program who have been such a large source of strength for me: Tiffany Youssef, Kelsie Smith, Sophie Winchester, Isabella Borth, Arsha Vuppuluri, Lauren Lindeen, Gigi Moody, Daniel Wiesner, Milap Naik, Salaar Ahmed, Ben Farnham, Alex Morrison, Cooper Cearns, Thomas Dlugosz, Mitch Donley, Michelle Hill, Dylan Bass, Fiona Hoffer, Emily Bossard, and to many more I wasn't able to name here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Embodied tasks that require active perception are key to improving language grounding models and creating holistic social agents. In this dissertation we explore four multi-modal embodied perception tasks and which require localization or navigation of an agent in an unknown temporal or 3D space with limited information about the environment. We first explore how an agent can be guided by language to navigate a temporal space using reinforcement learning in a similar way to that of a 3D space. Next, we explore how to teach an agent to navigate using only self-supervised learning from passive data. In this task we remove the complexity of language and explore a topological map and graph-network based strategy for navigation. We then present the WHERE ARE YOU? (WAY) dataset which contains over 6k dialogs of two humans performing a localization task. On top of this dataset we design three tasks which push the envelope of current visual language-grounding tasks by introducing a multi-agent set up in which agents are required to use active perception to communicate, navigate, and localize. We specifically focus on modeling one of these tasks, Localization from Embodied Dialog (LED). The LED task involves taking a natural language dialog of two agents – an observer and a locator – and predicting the location of the observer agent. We find that a topological graph map of the environments is a successful representation for modeling the complex relational structure of the dialog and observer locations. We validate our approach on several state of the art multi-modal baselines and show that a multi-modal transformer with large-scale pretraining outperforms all other models. We additionally introduce a novel analysis pipeline on this model for the LED and the Vision Language Navigation (VLN) task to diagnose and reveal limitations and failure modes of these types of models.

CHAPTER 1
INTRODUCTION

A main goal in artificial intelligence (AI) research is to develop systems that holistically understand and seamlessly interact with the world. This long-standing goal will only be achieved when agents can accurately perceive the world while reasoning about their perceptions to make actionable decisions. Taking it a step further, the seamless interaction of agents in the world requires agents that can communicate effectively, in natural language or otherwise, with both humans and other agents. Towards achieving these goals, challenging embodied tasks have been created which combine vision, language, and temporal decision-making.

However, we find many of the vision-language embodied tasks suffer from small annotated datasets and often over-fit to the small number environments available for the agents to be trained in. In this thesis we utilize passive data to pre-train and also fully train embodied agents for the embodied AI tasks of localization and navigation. Additionally we examine localization and navigation tasks over multiple levels of complexity of the language and posit that dialog is a more complex form of language that creates unique challenges during learning. This leads us to the central statement of this thesis that **embodied agents can learn to navigate and localize using dialog in unknown environments and in the face of limited scene information.**

We begin in Chapter 3 by addressing a language grounding localization task using RL, as a means to illustrate the strengths and weaknesses of the approach. Specifically, we address the task of Temporal Activity Localization via Language Query (TALL) in video. We show that the task of efficient localization of activities in videos is analogous to the task of embodied navigation and draw inspiration from work in language guided navigation to design a novel agent called *TripNet* [1]. TripNet navigates through a video looking only at

a small candidate window of frames and makes navigation actions of which video frames to look next based on its observation history and the description of the activity to be retrieved. We find that we can use a gated-attention architecture to create a rich cross-modal state representation and an actor-critic policy to determine the agent's actions. By creating an agent that is able to navigate the video in windows instead of processing the full video, we are able present an approach that is efficient (looks at only 40% of frames) while maintaining high accuracy. TripNet illustrates the effectiveness of RL as a solution approach. However, the efficiency of TripNet in practice stems from the fact that the agent only needs to interact with a video, not with a real-world environment. Specifically, TripNet's interactions involve seeking to specific point in the video and evaluating frames. In contrast, when RL is used for real-world agent learning, the cost of interaction is much higher due to the need to move and manipulate matter. This motivates our investigation of alternative approaches in Chapter 4.

In Chapter 4 we introduce the idea that navigation models can be learned without reinforcement learning (RL) and learnt over only passive data due to the fact that embodied navigation is a highly structured problem based on distance [2]. Specifically we propose a self-supervised approach to learn to navigate from only passive videos of agents roaming. Chapter 4 studies this idea in the context of image-goal navigation which is more simple test-bed for the approach. Our approach is simple and scalable, yet highly effective as it outperforms RL-based formulations by a significant margin. Our navigation agent builds and maintains a topological map of the environment and uses a sub-goal selection policy based on predicted distance-to-goal for each sub-goal. To model the distance-to-goal over explorable areas we present a graph network over the topological map, and show it is successful at propagated visual and semantic features. This approach demonstrates that successful embodied agents for navigation in novel environments can be trained without RL and without online interaction via learning semantic information about indoor environments via passive data.

In Chapter 5, we extend our focus to include embodied language-guided localization with multiple agents. We introduce WHERE ARE YOU? (WAY) [3] dataset which presents a unique set of challenges not yet present in other Embodied AI datasets [4, 5] or in other image-based conversational agents [6]. This dataset is composed of $\sim$6k dialogues in which two humans – an Observer and a Locator – complete a cooperative localization task. The Observer is spawned at random in a 3D environment and can navigate from first-person views while answering questions from the Locator. The Locator must localize the Observer in a detailed top-down map by asking questions and giving instructions. On top of the WAY dataset, we define three challenging localization based tasks: Localization from Embodied Dialog or LED (localizing the Observer from dialog history), Embodied Visual Dialog or EVD (modeling the Observer), and Cooperative Localization (modeling both agents). Successful agents in this cooperative task require goal-driven questioning based on the dialog history and theory of mind about the other agent, unambiguous answers communicating observations via language, and active perception and navigation to investigate the environment and seek out discriminative observations. This chapter focuses on the generation of the dataset, design of the tasks and creation of a simple baseline model for the first task, LED.

A significant challenge of the LED task is ensuring proper grounding of language in the visual domain and performing reasoning across the two modalities to make decisions of navigation (observer agent) and localization (locator agent). The environments contained in the WAY dataset can be represented as topological graphs, 3D-meshes or 2D top down images. In Chapter 5 we use 2D top down maps to model our baseline LED agent. Inspired by the success of graph networks for modeling distance for navigation, in Chapter 6 we transition to graph based map representations of the 3D environment. Over this environment we experiment with graph networks and popular multi-modal architectures. Specifically, we propose a viso-linguistic transformer, LED-Bert, for the Localization from Embodied Dialog task and instantiate a new version of the LED task which does local-

ization over the navigation graph. We demonstrate a pre-training schema for LED-BERT which utilizes large scale web-data as well as other multi-modal embodied AI task data to learn the visual grounding required for successful localization's in LED. We show LED-Bert is able to achieve SOTA performance and outperform other learned baselines by a significant margin.

The LED-Bert model and pre-training schema is inspired by recent success in transfer learning for multi-modal problems ranging from visual dialog [7] to the embodied task of Vision Language Navigation (VLN) [8]. In Chapter 7, we dive into evaluation of transformer based VLN models and the LED-Bert model. In this chapter we specifically seek to diagnose what may attribute to success and failure modes in regards to how the model is attending to different types of tokens in the dialog or navigation instructions. Pre-training schemes help these models learn visual grounding for objects and object attributes. Unlike static multi-modal perception problems, such as visual question answering or image captioning, multi-modal embodied perception tasks additionally require action grounding. For example the ability to ground the instruction "take a left at the couch" into the turning left action.

We specifically wanted to investigate the models ability to perform action grounding. Additionally we were motivated to do this analysis via a wish to understand how using panoramic nodes and environments may effect the spatial understanding of the ViLBert based VLN and LED models. In our analysis, we devise a simple token masking experiment at inference time, which masks out different token types and then measures the impact of masking on performance. We argue that if masking doesn't affect accuracy significantly than the model does not significantly utilize that token type when making predictions. We surprisingly find a severe limitation of these models which is that they almost exclusively rely on noun tokens, meaning that the models seem to only rely on the objects in the path and not the directional information of the path. In this chapter we investigate two pre-training strategies and one data augmentation method which we find increases overall task

performance as well as slightly increases these transformer based discriminative VLN models ability to ground actions from instructions and consequently increase performance of the model. Through our analysis and efforts to increase model performance, we posit that using a discriminative model approach to the VLN will always be more prone to this limitation, however this also highlights a area for future progress on VLN models. Additionally, efforts for new models for VLN should include token analysis to understand model behavior.

CHAPTER 2
BACKGROUND

2.1 Embodied AI

The field of Embodied Artificial Intelligence (Embodied AI or EAI) studies AI agents in physical spaces. Agents in EAI can have the capabilities of moving, seeing, speaking or manipulating objects in their physical world. EAI research is focused on designing agents that can intelligently use these capabilities to reach their goal objective. The field of EAI has grown significantly in recent years, which can be in many ways attributed to the growing abundance of fast and convenient simulators and realistic 3D datasets. This has allowed a test bed for virtual embodied agents, which can be easily trained and tested on specific capabilities. The research community has generated a variety of EAI tasks to be tested in these simulated environments, with the hopeful promise of future transfer to robots operating in real world environments. EAI is often characterized by an agent with a first person field of view (FP-FOV), the notion of a body or the ability to take actions that change its own observations. The EAI tasks we discuss in this thesis operate under the condition of testing on a set of unseen environments, therefore evaluating the ability of an agent to generalize its abilities to new environments.

Prior work can be organized into three broad topics. The first topic is language and vision tasks (section 2.2), which is relevant to activity localization (chapter 3), our novel embodied dialog dataset (chapter 5), and our proposed work in chapter 6. The second topic is 3D navigation (section 2.3), which provides prior art for our work on navigation from passive data in chapter 4. The third topic is graph neural networks (section 2.4), which are used to model navigational distance in chapter 4 and are proposed to model localization via dialog in chapter 6.

2.2 Language and Vision Tasks

2.2.1 Embodied Language Tasks.

A number of 'Embodied AI' tasks that require language, visual perception, and navigation in realistic 3D environments have recently gained prominence, including Embodied Question Answering [9, 10], instruction based navigation [4, 11, 12, 13, 14], and challenges based on household tasks [15, 16]. These embodied tasks involve active perception – an agent predicting navigation actions from language and visual observations. Several papers have also extended the VLN task – in which an agent must follow natural language instructions to traverse a path in the environment – to multi-agent and dialog settings. Nguyen and Daumé III [17] consider a scenario in which the agent can query an oracle for help while completing the navigation task. Cooperative Vision-and-Dialog Navigation (CVDN) [5] is a dataset of dialogs in which a human assistant, with access to visual observations from an oracle planner, helps another human complete a navigation task. CVDN dialogs are set in the Matterport3D buildings [18] and critically they are goal-oriented and easily evaluated.

2.2.2 Image-based Dialog

Several datasets grounding goal-oriented dialog in natural images have been proposed. The most similar settings to the dataset and tasks we present, are Cooperative Visual Dialog [6, 19], in which a question agent (Q-bot) attempts to guess which image from a provided set the answer agent (A-bot) is looking at, and GuessWhat?! [20], in which the state estimation problem is to locate an unknown object in the image. Our dataset extends these settings to a situated 3D environment allowing for active perception and navigation on behalf of the A-bot (Observer), and offering a whole-building state space for the Q-bot (Locator) to reason about.

2.2.3 3D Localization from Language.

While localization from dialog has not been intensively studied, embodied localization from language has been studied as a sub-component of instruction-following navigation agents [21, 22, 23]. The LingUnet model – a generic language-conditioned image-to-image network we use as the basis of our LED model in section 5.4 – was first proposed in the context of predicting visual goals in images [24]. This also illustrates the somewhat close connection between grounding language to a map and grounding referring expressions to an image [25, 26]. It is important to note that localization is often a precursor to navigation – one which has not been addressed in existing work in language-based navigation. In both VLN and CVDN, the instructions are conditioned on specific start locations – assuming the speaker knows the navigator's location prior to giving directions. The localization tasks that we present in this thesis on the WAY dataset fill this gap by introducing a dialog-based means to localize the navigator. This requires capabilities such as describing a scene, answering questions, and reasoning about how discriminative potential statements will be to the other agent.

2.2.4 Temporal Localization from Language.

There has also been significant work in temporal localization in the visual domain using language. In the Temporal Activity Localization via Language query (TALL) task an AI agent is given a video and text description of an activity, and the agent must select the appropriate temporal boundaries of the video clip containing the described activity. The TALL task was introduced by [27, 28]. These works additionally introduced a dataset for the task, Charades-STA [27] and DiDeMo [28]. Each dataset contains untrimmed videos with multiple sentence queries and the corresponding start and end timestamp of the clip within the video. The two papers adopt a supervised cross modal embedding approach, in which sentences and videos are projected into a embedding space, optimized so that queries and their corresponding video clips will lie close together, while non-corresponding clips

will be far apart. At test time both approaches run a sliding window across the video and compute an alignment score between the candidate window and the language query. The window with the highest score is then selected as the localized query. Follow-up works on the TALL task have differed in the design of the embedding process [29]. Multiple works [30, 31, 32] modified the original approach by adding self-attention and co-attention to the embedding process. Xu, He, Sigal, Sclaroff, and Saenko [33] introduce early fusion of the text queries and video features rather than using an attention mechanism. Additionally, [33] uses the text to produce activity segment proposals as their candidate windows instead of using a fixed sliding window approach. Preprocessing of the sentence queries has not been a primary focus of previous works, with most methods using a simple LSTM based architecture for sentence embedding, with Glove [34] or Skip-Thought [35] vectors to represent the sentences. Video-based localization that predates TALL either uses a limited set of text vocabulary to search for specific events or uses structured videos [36].

2.2.5 Temporal Activity Localization.

Temporal action localization refers to localizing activities over a known set of action labels in untrimmed videos. Some existing work in this area has found success by extracting CNN features from the video frames, pooling the features and feeding them into either single or multi-stage classifiers to obtain action predictions, along with temporal labels [37, 38, 39, 40]. These methods use a sliding window approach. Often, multiple window sizes are used and candidate windows are densely sampled, meaning that the candidates overlap with each other. This method has high accuracy, but it is an exhaustive search method that leads to high computational costs. There are also TALL methods that forgo the end-to-end approach and instead use a two-stage approach: first generating temporal proposals, and second, classifying the proposals into action categories [41, 42, 43]. Unlike both sets of previous methods, our proposed model TripNet, uses reinforcement learning to perform temporal localization using natural language queries. In action-recognition methods [44,

45, 46], Frame glimpse [47] and Action Search [48] use reinforcement learning to identify an action while looking at the smallest possible number of frames. These works focus only on classifying actions (as opposed to parsing natural language activities) in trimmed videos and and the former does not perform any type of temporal boundary localization. A recent work [49] also uses RL to learn the temporal video boundaries. However, it analyzes the entire video and does not focus on the efficiency of localizing the action. Another work [50], uses RL to estimate the activity location using fixed temporal boundaries. Furthermore, it uses Faster R-CNN trained on the Visual Genome dataset to provide semantic concepts to the model. Instead, we use gated attention over C3D features to learn semantic concepts which is more efficient. Furthermore, here all frames are used as an input which can improve the accuracy performance but requires all features to be extracted. However, we provide feature extraction on-demand and are more efficient.

2.3 3D Navigation

2.3.1 Navigation in simulators.

Navigating tasks largely fall into two main categories [51], ones in which a goal location is known [52, 53, 54] and limited exploration is required and ones which the goal location is not known and efficient exploration is necessary. In the second category, tasks range from finding the location of specific objects [55], rooms [56], or images [57] to the task of exploration itself [58]. The majority of current work [53, 57, 59, 60] leverages simulators [61] and extensive interaction to learn end-to-end models models for these tasks. In this work we will show that the semantic cues that are needed for exploration based navigation tasks can be learned simply from viewing video trajectories.

2.3.2 Navigation using passive data.

Several prior works have tackled the navigation tasks when there is some passive experience available in the test environment [62, 63, 64, 65, 66, 67]. A more limited number of

works train navigation policies without simulation environments [63, 66]. Unlike these works, we tackle the task of image goal navigation in unseen environments where there is no experience available to the agent in the test environments during training. Chaplot, Salakhutdinov, Gupta, and Gupta [68] and Chang, Gupta, and Gupta [69] learn navigation from passive data without having access to any experience in the test environment. The work presented in this chapter is closely related to Neural Topological SLAM (NTS) [68] which builds a topological map and estimates distance to the target image using a learned function. Unlike our method, NTS requires access to panoramic observations and ground-truth maps for training the distance function. This makes our method much more scalable as it can be trained with just video data with arbitrary field of view. Chang, Gupta, and Gupta [69] also use a similar approach for object goal navigation, while also incorporating video data from Youtube for learning a Q function. A key difference is that our method learns a episodic distance function, utilizing all past observations to estimate distances to the target image. In comparison, Chaplot, Salakhutdinov, Gupta, and Gupta [68] and Chang, Gupta, and Gupta [69] use a memory-less distance function operating only on the agent's current observation.

2.3.3 Map Based Navigation.

There are multiple spatial representations which can be leveraged in navigation tasks. Metric maps, which maintain precise information of occupied space in an environment, are commonly used for visual navigation tasks [58, 70, 71]. Metric maps, however, suffer from issues of scalability and consistency under sensor noise. Topological maps however have recently gained more traction as navigational maps [68, 65, 69, 67] to combat these issues. Additionally, topological maps for robotic navigation draw inspiration from both animal and human psychology. The cognitive map hypothesis proposes that the brain builds coarse internal spatial representations of environments [72, 73]. Multiple works argue this internal representation relies on landmarks [74, 75] making human cognitive maps more

similar to the topological maps as opposed to rigid metric maps.

2.4 Graph Neural Networks

Graph Neural Networks (GNN) are neural networks that operate over a graph structure and are used for modeling relational data in the non-euclidean space. A graph is composed of a set of nodes and edges. Nodes and edges can contain attributes that are passed as inputs to a graph network. GNNs are specifically used for modeling relational data in the non-euclidean space and have shown success on a variety of applications, such as, modeling physical robotic systems. GNN architectures can take spectral or spatial approaches to aggregate information across the graph. The networks we examine in this thesis for localization and navigation use a spatial approach, which define the convolutions based on the graph topology and local neighborhood.

2.4.1 Graph Networks on Topological Maps

In the tasks of indoor navigation and localization, a graph structure naturally arises from the topological maps of the 3D environments and therefore we find GNNs are well-suited to model these tasks. Many works in the navigation space use topological maps to represent the environment [76, 65, 67, 69, 56]. Topological maps of environments can be built and maintained in different ways. Some agents are given a structured graph [76, 65], some build the map via an exploration or walk through phase [67] and some build the map as they navigate to the goal state [68]. GNNs have been successfully used over these topological map representations for robotic localization as a module of navigation algorithms [65, 67] using simple Graph Convolutional Networks.

2.4.2 Language Conditioned Graph Networks

GNNs haven been proven to work for the multi-model language and vision tasks for tasks such as video and image captioning [77, 78] and visual question answering (VQA) [79].

For these tasks there is not a specific way to represent the task as a graph so multiple approaches have been explored. To model the VQA task, Teney, Liu, and Den Hengel [79] first created a scene graph from the input image and a syntactic graph from the input question and then combined the graphs as input to a GGNN which train embeddings for predicting the question's answer. Alternately for the same task, Norcliffe-Brown, Vafeias, and Parisot [80] builds the relational image graphs as conditioned on the question. In recent work GNNs have been extended to the multi-model Vision and Language Navigation task [81, 82]. Chen, Chen, Chuang, Vázquez, and Savarese [81] uses a GNN to compute information about connectivity and visual features in the graph before using a transformer to combine the language modality with the environment information. In contrast, Hong, Rodriguez-Opazo, Qi, Wu, and Gould [82] creates a language conditioned visual graph at each time step using the agents current state. This visual graph is then used to predict the next navigational action. Both works demonstrate the success of GNNs in the multi-modality setting on top of a navigation task, giving inspiration for the use of GNNs on the LED and EVD tasks.

2.5 Transformers.

2.5.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) [83] is a transformer based encoder used for language modeling. BERT is trained on massive amounts of unlabeled text data, and takes as input sentences of tokenized words and corresponding positional embeddings per tokens. BERT is trained using the masked language modeling and next sentence prediction training objectives. In the masked language modeling schema, 15% of the input tokens are replaced with a [MASK] token. The model is then trained to predict the true value of the input tokens which are masked using the other tokens as context. In the next sentence prediction schema, the model is trained to predicted if the two input sentences follow each other or not. BERT is specifically trained on Wikipedia and

BooksCorpus [84].

2.5.2 Vision-and-Language Pretraining

Recently multiple works have experimented with utilizing dual-stream transformer based models that have been pretrained with self-supervised objectives and transferring them to downstream multi-modal tasks with large success. This has been seen for tasks such as Visual Question Answering [85], Commonsense Reasoning [86], Natural Language Visual Reasoning [87], Image-Text Retrieval [88], and Visual-Dialog [7] and Vision Language Navigation [8]. Specifically VLN-BERT and VisDial + BERT adapt the ViLBERT architecture and utilize a very similar pretraining schema to that of LED-BERT.

CHAPTER 3

LANGUAGE GUIDED LOCALIZATION OF ACTIVITIES IN VIDEO

3.1 Introduction

In this chapter we introduce the first localization task to be considered in this dissertation and a novel RL-based solution approach. The contents of this chapter appeared in [1]. The Temporal Activity Localization via Language query (TALL) task [27, 28], illustrated in Figure 3.1, provides an AI agent with a video and text description of an activity, and requires the agent to select the appropriate temporal boundaries of the video clip containing the described activity. This task is challenging because it requires effective language grounding in the temporal visual modality to discard irrelevant clips of the video and fine grained localization of the precise start and end of activities. Motivated by the increasing availability of videos and their importance in application domains, such as social media and surveillance, there is a pressing need for automated video analysis methods. It is natural for humans to use natural language to describe events and therefore there is a need to create video analysis methods that operate over these descriptions. While classical video retrieval works at the level of entire clips, a more challenging and important task is to efficiently sift through large amounts of unorganized video content and retrieve specific moments of interest. In order to develop an efficient solution to the TALL task it is necessary to address two main challenges: 1) Devising an effective joint representation (embedding) of video and language features to support localization; and 2) Learning an efficient search strategy to sample a video clip intelligently to localize an event of interest.

Many previous works [30, 31, 32] focus on the first challenge and present multi-modal architectures which obtain encouraging results, however as they ignore the second challenge, they all suffer from a significant limitation of scalability and efficiency. Many of

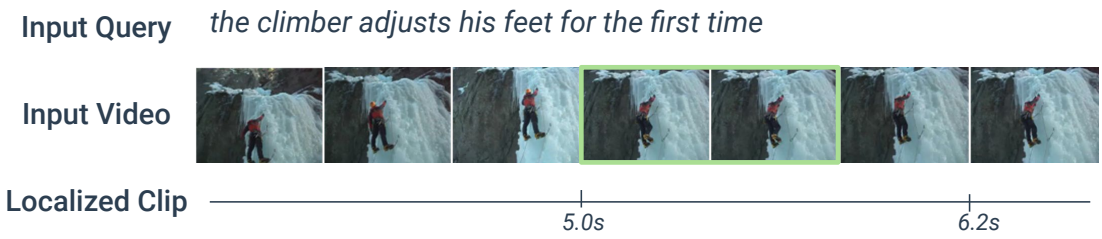

Figure 3.1: TALL task: an AI agent is given a video and text description of an activity. The agent must localize the temporal bounds of video clip containing the described activity. The green bounding denotes the desired output, the first time in which the feet are being adjusted and 'Localized Clip' denotes the output of the agent. Note that solving this problem requires local and global temporal information: local cues are needed to identify frames in which the feet are adjusted and global cues are also needed to identify the first occurrence of the activity, since the climber adjusts his feet throughout the video.

these methods construct temporally-dense representations of the entire video clip which are then analyzed to identify the target events. This methodology does not efficiently scale to longer videos where, where the target might be a single short moment. We can compare this dense sampling approach to how humans search events of interest. A human would fast-forward through the video from the beginning, effectively sampling sparse sets of frames until they got closer to the region of interest. Then, they would look frame-by-frame until the start and end points are localized. At that point the search would terminate, leaving the vast majority of frames unexamined. Note that efficient solutions must go well beyond simple heuristics, since events of interest can occur anywhere within a target clip and the global position may not be obvious from the query.

Locating a specific activity within a video using an agent that mimics a human search strategy is analogous to the task of navigating through a three dimensional world. How can we efficiently learn to navigate through the temporal landscape of the video? The parallels between our approach and navigation lead us to additionally review the related work in the areas of language-based navigation and embodied perception. For example, in the task of Embodied Question Answering [9] an agent is asked questions about the environment, such as "what color is the bathtub," and the agent must navigate to the bathtub and answer the question. Methods to solve this task focus on grounding the language of the question

not directly into the pixels of the scene but into actions for navigating around the scene. Predicting navigation actions from language and visual observations is seen in a variety of embodied tasks [4, 14, 9]. Similarly our presented approach grounds the text query into actions for skipping around the video to narrow down on the correct clip.

We present an approach, TripNet, to learning an efficient search strategy that can mimic the human ability to intelligently navigate a video to localize a specific moment. Additionally, we show that using a gated-attention architecture is effective in aligning the text queries, which often consist of an object and its attributes or an action, with the video features that consist of convolutional filters that can identify these elements. Two prior works [32, 33] also utilize an attention model, but they do not address its use in efficient search. We address the challenge of efficient search through a combination of reinforcement learning (RL) and fine-grained video analysis.

Our approach to temporal localization uses a novel architecture for combining the multi-modal video and text features with a policy learning module that learns to step forward and rewind the video and receives awards for accurate temporal localization. Note that in contrast to RL in language-guided navigation tasks in 3D environments [4, 14, 9, 10], TripNet does not need to learn a physical local navigation policy since it is not operating in a 3D space. Therefore online interaction to learn things such as obstacle avoidance is not necessary. Additionally there is no navigation cost of moving between far away frames, unlike moving between points in 3D, so the focus of the agent is simply to bring the agent closer to the target window. In summary we make two contributions: First, we present a novel end-to-end reinforcement learning framework called TripNet that addresses the problem of temporal activity localization via a language query. TripNet uses gated-attention to align text and visual features, leading to improved accuracy. Second, we present experimental results on the datasets Charades-STA [27], ActivityNet Captions [89] and TACoS [90]. These results demonstrate that TripNet achieved state of the art accuracy results in 2019 when the experiments were performed. In addition TripNet significantly

increases efficiency, by evaluating only 32-41% of the total video.

3.2 TripNet Architecture

We now describe our approach TripNet, an end to end reinforcement learning method for localizing and retrieving temporal activities in videos given a natural language query as shown in Figure 3.2.

Problem Formulation. The localization problem that we solve is defined as follows: Given an untrimmed video V and a language query L, our goal is to temporally localize the specific clip W in V which is described by L. In other words, let us denote the untrimmed video as $V = \{f_n\}_{n=1}^{N}$ where N is the number of frames in the video, we want to find the clip $W = \{f_n, \ldots, f_{n+k}\}$ that corresponds best to L. It is possible to solve this problem efficiently because videos have an inherent temporal structure, such that an observation made at frame n conveys information about frames both in the past and in the future. Some challenges of the problem are, how to encode the uncertainty in the location of the target event in a video, and how to update the uncertainty from successive observations. While a Bayesian formulation could be employed, the measurement and update model would need to be learned and supervision for this is not available.

Since it is computationally feasible to simulate the search process (in fact it is only a one-dimensional space, in contrast to standard navigation tasks) we adopt an reinforcement learning (RL) approach. We are motivated by human annotators who observe a short clip and make a decision to skip forward or backward in the video by some number of frames, until they can narrow down to the target clip. We emulate this sequential decision process using RL. Using RL we train an agent that can steer a fixed sized window around the video to find W without looking at all frames of V. We employ the actor-critic method A3C [91] to learn the policy π that maps $(V, L) \rightarrow W$. The intuition is that the agent will take large jumps around the video until it finds visual features that identify proximity to L, and then it will start to take smaller steps as it narrows in on the target clip.

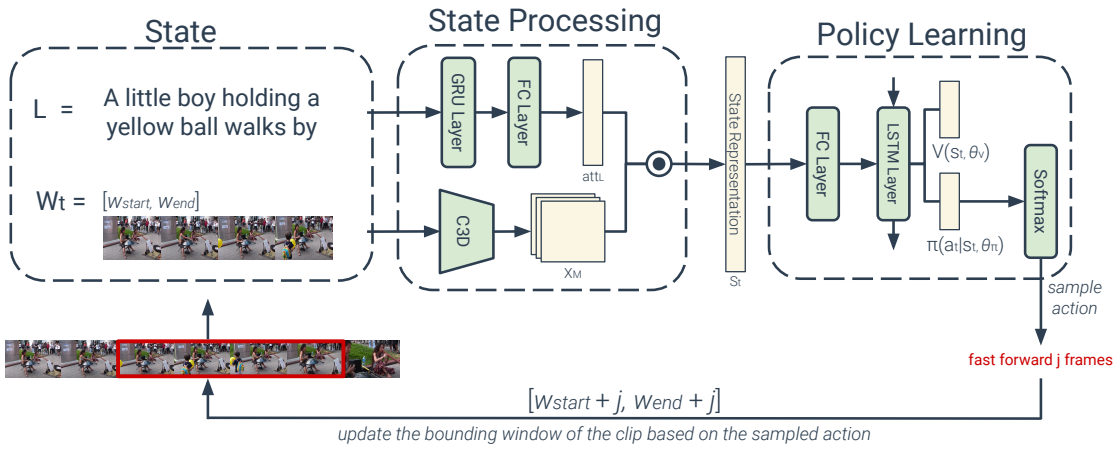

Figure 3.2: TripNet Architecture: consists of a state processing module and a policy learning module. A state consists of the language query L, the frames of the current window F_{Wt} and the location of the window in the video Wt. The state processing module encodes the state into a joint visual and linguistic representation. This representation is passed into the policy learning module learns the action policy and value function. The action with the highest probability is sampled via a softmax as shown in red and the state is updated.

3.2.1 State and Action Space

At each time step, the agent observes the current state, which consists of the sentence L and a candidate clip of the video. The clip is defined by a bounding window $[W_{start}, W_{end}]$ where start and end are frame numbers. At time step $t = 0$, the bounding window is set to $[0, X]$, where X is the average length of annotated clips within the dataset.[1] This window size is fixed and does not change. At each time step the state processing module creates a state representation vector which is fed into the policy learning module, on which it generates an action policy. This policy is a distribution over all the possible actions. An action is then sampled according to the policy.

Our action space consists of 7 predefined actions: move the entire bounding window W_{start}, W_{end} forward or backward by h frames, j frames, or 1 second of frames or TERMINATE. Where $h = N/10$ and $j = N/5$. These navigation steps makes aligning the

[1]Note that this means that the search process always starts at the beginning of the video. We also explored starting at the middle and end of the clip and found empirically that starting at the beginning performed the best.

visual and text features easier. Making the RL agent explicitly learn the window size significantly increases the state space and leads to a drop in accuracy and efficiency with our current framework. However, an additional function to adjust the box width over the fixed window size could be learned, analogous to the refinement of the anchor boxes used in object detectors. These actions were chosen so that amount of movement is proportional to the length of the video. If the bounding window is at the start or end of the full video and an action is chosen that would push the bounding window outside the video's length, the bounding window remains the same as the previous time step. The action TERMINATE ends the search and returns the clip of the current state as the video clip predicted to be best matched to L.

3.2.2 TripNet architecture

We now describe the architecture of TripNet, illustrated in Figure 3.2.

State Processing Module. At each time step, the state-processing module takes the current state as an input and outputs a joint-representation of the input video clip and the sentence query L. The joint representation is used by the policy learner to create an action policy from which the optimal action to take is sampled. The clip is fed into C3D [92] to extract the spatio-temporal features from the fifth convolutional layer. We mean-pool the C3D features across frames and denote the result as x_M.

To encode the sentence query, we first pass L through a Gated Recurrent Unit (GRU) [93] which outputs a vector x_L. We then transform the query embedding into an attention vector that we can apply to the video embedding. To do so, the sentence query embedding x_L is passed through a fully connected linear layer with a sigmoid activation function. The output of this layer is expanded to be the same dimension of x_M. We call the output of the linear layer the attention vector, att_L. We perform a Hadamard multiplication between att_L and x_M and the result is output as the state representation s_t. This attention unit is our gated attention architecture for activity localization. Hence, the gated attention unit is designed to

gate specific filters based on the attention vector from the language query [94]. The gating mechanism allows the model to focus on specific filters that can attend to specific objects, and their attributes, from the query description.

To demonstrate the effectiveness of our gated-attention architecture, we implement an additional baseline called TripNet-Concat which does a simple concatenation operation between the video and text representations. In TripNet-Concat, self-attention is only performed over the mean pooled C3D features and a Skip-Thought [35] encoding of the sentence query is concatenated with the features of the video frames to produce the state representation. In this baseline only the state processing module changes and the policy learning module remains the same.

Policy Learning Module. We use an actor-critic method to model the sequential decision process of grounding the language query to a temporal video location. The module employs a deep neural network to learn the policy and value functions. The network consists of a fully connected linear layer followed by an LSTM, which is followed by a fully connected layer to output the value function $v(s_t|\theta_v)$ and fully connected layer to output the policy $\pi(a_t|s_t, \theta_\pi)$, for state, s_t and action, a_t at time t, θ_v is the critic branch parameters and θ_π is actor branch parameters. The policy, π, is a probabilistic distribution over all possible actions given the current state. Since we are trying to model a sequential problem we use an LSTM so that the system can have memory of the previous states which will inevitably positively impact the future actions. Specifically, we use the asynchronous actor-critic method known as A3C [91] with Generalized Advantage Estimation [95] that reduces policy gradient variance. The method runs multiple parallel threads that each run their own episodes and update global network parameters at the end of the episode.

Since the goal is to learn a policy that returns the best matching clip, we want to reward actions that bring the bounding windows [W_{start}, W_{end}] closer to the bounds of the ground truth clip. Hence, the action to take, should return a state that has a clip with more overlap with the ground-truth than the previous state. Therefore, we use reward shaping by having

our reward be the difference of potentials between the previous state and current state. However, we want to ensure the agent is taking an efficient number of jumps and not excessively sampling the clip. In order to encourage this behavior, we give a small negative reward in proportion with the total number of steps thus far. As a result, the agent is encouraged to find the clip window as quickly as possible. We experiment to find the optimal negative reward factor β. We found using a negative reward factor results in the agent taking more actions with larger frame jump. Hence, our reward at any time step t is calculated as follows:

$$reward_t = (IOU_t(s_t) - IOU_t(s_{t-1})) - \beta * t \tag{3.1}$$

where we set β to .01. We calculate the IOU between the clip of the state at time t, [W^t_{start}, W^t_{end}], and the ground truth clip for sentence L, [G_{start}, G_{end}] as follows:

$$IOU_t = \frac{min(W^t_{end}, G_{end}) - max(W^t_{start}, G_{start})}{max(W^t_{end}, G_{end}) - min(W^t_{start}, G_{start})} \tag{3.2}$$

We use the common loss functions for A3C for the value and policy loss. For training the value function, we set the value loss to the mean squared loss between the discounted reward sum and the estimated value:

$$Loss_{value} = \sum_t (R_t - v(s_t|\theta_v))^2 * \gamma_1, \tag{3.3}$$

where we set γ_1 to .5 and R_t is the accumulated reward. For training the policy function, we use the policy gradient loss:

$$\begin{aligned} Loss_{policy} = -\sum_t log(\pi(a_t|s_t, \theta_\pi)) * GAE(s_t) \\ -\gamma_0 * H(\pi(a_t|s_t, \theta_\pi)), \end{aligned} \tag{3.4}$$

where GAE is the generalized advantage estimation function, H is the calculation of entropy

and γ_0 is set to 0.5. Therefore, the total loss for our policy learning module is:

$$Loss = Loss_{\pi} + \gamma_1 + Loss_v. \tag{3.5}$$

3.3 Implementation details.

During training, we take a video and a single query sentence that has a ground truth temporal alignment in the clip. At time $t = 0$ we set the bounding window [W_{start}, W_{end}] to be [0, X] where X is the average length of ground truth clips in the dataset. This means that this is the initial clip in the sequential decision process. Furthermore, it also means that the first actions selected will most likely be skipping forward in the video. The input to the system is X sequential video frames and a sentence query. The sentence is first encoded through a Gated Recurrent Unit of size 256 and then through a fully-connected linear layer of size 512 with sigmoid activation. We run the video frames within the bounding window through a 3D-CNN [46] which is pre-trained on the Sports-1M dataset and extract the 5th convolution layer. The A3C reinforcement learning method is then used for the policy learning module and is trained with stochastic gradient descent (SGD) with a learning rate of $.0005$. The first fully-connected (FC) layer of the policy learning module is 256 dimensions and is followed by an long short term memory (LSTM) layer of size 256. During training, we set A3C to run 8 parallel threads.

3.4 Experimental Setup

We evaluate the TripNet architecture on three video datasets, Charades-STA [27], ActivityNet Captions [89] and TACoS [96]. Charades-STA was created for the moment retrieval task and the other datasets were created for video captioning, but are often used to evaluate the moment retrieval task. Note that we chose not to include the DiDeMo [28] dataset because the evaluation is based on splitting the video into 21 pre-defined segments, instead of utilizing continuously-variable start and end times. This would require a change in the set

of actions for our agent. We do, however, compare our approach against the method from Hendricks, Wang, Shechtman, Sivic, Darrell, and Russell [28] on other datasets.

Evaluation Metric. In Table 3.1, we report Intersection over Union (IoU) at different alpha thresholds to measure the difference between the ground truth clip and the predicted clip. A prediction is classified as correct if the predicted clip and the ground truth clip have an IoU that is above the set alpha threshold. Refer to Equation 3.2 for how the IoU is calculated. Note all reported IoU scores are measured at R@1.

Comparison and Baseline Methods. We compare against prior work on the TALL task [28, 27, 29, 32, 30, 33, 31, 50] and a baseline version of the TripNet architecture. Aside from SM-RL, all prior works tackle the task by learning to jointly represent the ground truth moment clip and the moment description query. To generate candidate windows during testing, these methods go over the whole video using a sliding window and then, choose the candidate window that best corresponds to the query encoding. This methodology relies on seeing all frames of the video at least once, if not more, during test time.

3.5 Results

TripNet Outperforms Most Baselines. Table Table 3.1 shows results for the TALL task across 3 datasets, compiling the performance numbers reported in the prior works for comparison to the performance of TripNet. Note that these experiments were performed in 2019 and compared against works prior to this date. In terms of accuracy, TripNet outperforms all other methods on the Charades-STA and TACoS datasets, and that it performs comparably to the state of the art on ActivityNet Captions. Using a state processing module for TripNet that does not use attention (TripNet-Concat) performs consistently worse than the state processing module for TripNet that uses the gated attention architecture (TripNet-GA), since multi-modal fusion between vision and language shows improvement with different attention mechanisms. Our results show that a temporal localization method does not necessarily need to see the entire clip to achieve success. ABLR processes all frames and encodes

Table 3.1: The accuracy of each method on the TALL task across the datasets of ActivityNet-Captions, TACoS and Charades-STA. Accuracy is reporting using IoU at multiple α values at Rank@1.

	ActivityNet			TACoS			Charades		
Method	α@.3	α@.5	α@.7	α@.3	α@.5	α@.7	α@.3	α@.5	α@.7
CTRL [27]	28.70	14.00	-	18.32	13.3	-	-	23.63	8.89
MCN [28]	21.37	9.58	-	1.64	1.25	-	-	17.46	8.01
ABLR [32]	**55.67**	**36.79**	19.50	9.40	-	-	-	24.36	9.01
MLVI [33]	45.30	27.70	13.60	52.13	33.26	13.43	**54.70**	35.60	15.80
TGN [29]	45.51	28.47	-	21.77	18.9	-	-	-	-
ACRN [30]	31.29	16.17	-	19.52	14.62	-	-	-	-
VAL [31]	-	-	-	19.76	14.74	-	-	23.12	9.16
SM-RL [50]	-	-	-	20.25	15.95	-	24.36	11.17	-
TripNet-Concat	36.75	25.64	10.25	18.24	14.16	6.47	41.84	27.23	12.62
TripNet-GA	48.42	32.19	**13.93**	**23.95**	**19.17**	**9.52**	54.64	**38.29**	**16.07**

the intermediate representations of each frame and word using Bi-LSTM, followed by two levels of attention over it. It then regresses the coordinates over this comprehensive intermediate representation. Hence, this regression-based approach over all frames is beneficial for the TALL task. However, it is computationally inefficient to perform this processing over all frames. On the other hand, in our method feature extraction is only performed if requested by the RL step and is suitable for long surveillance videos where ABLR may not be practical. In Figure Figure 3.3, we show the qualitative results of TripNet-GA on the Charades-STA dataset. In the figure, we show the sequential list of actions the agent takes in order to temporally localize a moment in the video. We show two examples where the TripNet agent navigates through the video to observe different candidate windows before terminating the search.

Failure cases of TripNet. There is an overall drop in performance of TripNet on TACoS as seen in Table 3.1, despite outperforming other methods. We attribute this to TACoS being a challenging dataset because it contains long cooking videos set in a single kitchen scene and fine grained actions. We observe that TripNet is able to maintain high accuracy on ActivityNet in comparison to the Charades-STA dataset despite ActivityNet videos being

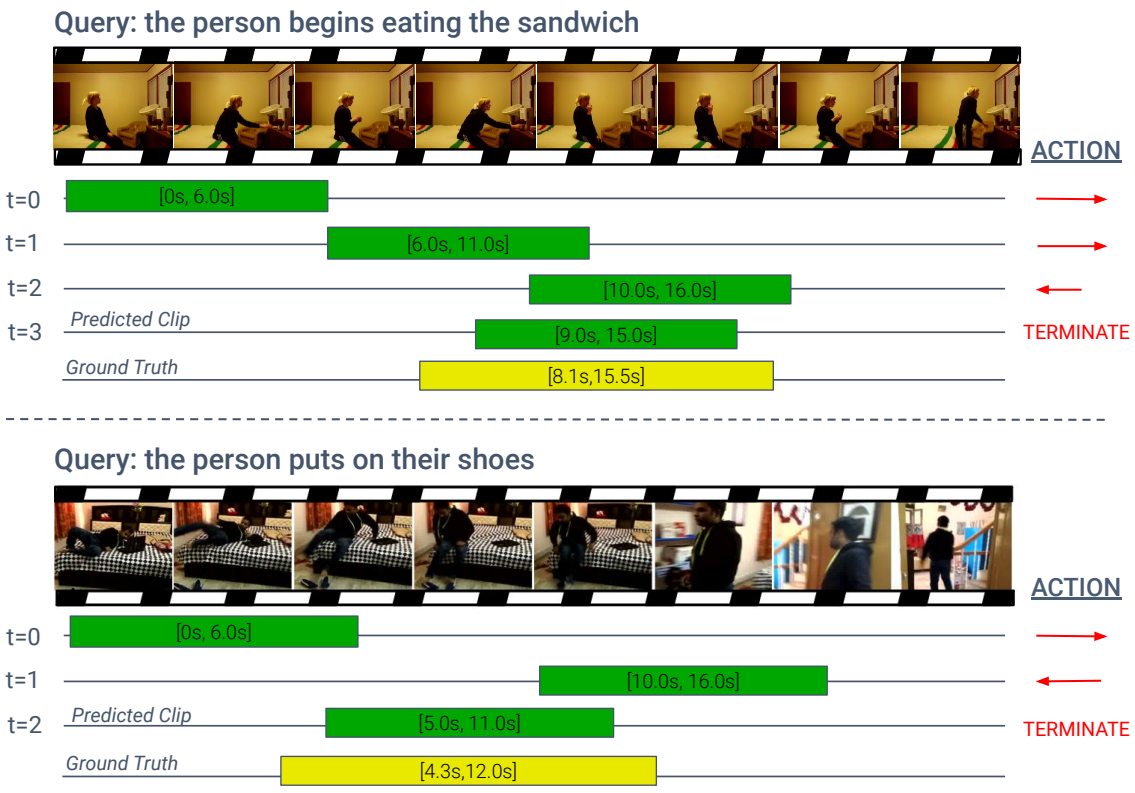

Figure 3.3: Qualitative results of TripNet-GA on the Charades-STA dataset. The green boxes represent the bounding window of the state at time t and the yellow box represents the ground truth bounding window. The first video is 33 seconds long (792 frames) and the second video is 20 seconds long (480 frames). In both examples the agent navigates both backwards and forwards in the video. In the first example, TripNet sees 408 of the frames of the video, which is 51% of the video. In the second example TripNet sees 384 frames of the video, which is 80% of the frames.

up to four times longer than Charades-STA videos. This illustrates TripNet's ability to scale to videos of different lengths. During qualitative analysis, we found that a large source of IoU inaccuracy for Tripnet was the size of the bounding window. This problem can be seen in the second example of results in Figure 3.3. TripNet currently uses a fixed size bounding window that the agent moves around the video until it returns a predicted clip. The size of the fixed window is equal to the mean length of the ground truth annotated clips. A possible direction for future work would be to add actions that expand and contract the size of the bounding window.

TripNet outperforms all baselines in terms of efficiency. TripNet is the only method

Table 3.2: Efficiency of the TripNet architecture on the TALL task across different datasets reported by number of frames seen, average actions taken, and the bounding window size used during inference. Note that the number of actions includes the TERMINATE action and th size of the bounding window is reported in terms of seconds. All experiments were performed on the same Titan Xp GPU.

Dataset	% of Frames Used	Avg. # Actions	Bounding Window Size
ActivityNet	41.65	5.56	35.7s
Charades	33.11	4.16	8.3s
TACoS	32.7	9.84	8.0s

Table 3.3: Comparison of Efficiency on the TALL task. Efficiency is report by the average time in milliseconds that it takes to localize a moment on different datasets. All methods were tested on the same Titan Xp GPU.

Method	Charades	ActivityNet	TACoS
CTRL [27]	44.19ms	218.95ms	342.12ms
MCN [28]	78.16ms	499.32ms	674.01ms
TGN [29]	18.2ms	90.2ms	144.7ms
TripNet-GA	5.13ms	6.23ms	11.27ms

that does not need to watch the entire video to temporally localize a described moment. Instead, our trained agent efficiently moves a candidate window around the video until it localizes the described clip. Measurements of efficiency are described in Table 3.2. Table 3.3 demonstrates the difference in efficiency between methods by showing the average localization time for TripNet-GA and some of the baseline methods. We find that TripNet outperforms all other methods and we see the largest computation gains on the TACoS dataset which contains the longest video clips.

3.6 Conclusions

In this chapter we show that we can borrow approaches from navigation tasks and apply it to the TALL task with high accuracy while only viewing a fraction of the frames. We learn an agent via reinforcement learning to intelligently navigate through long, untrimmed

videos guided by the video features and a natural language query to localize a specific moment. The model we present uses gated-attention mechanism over cross-modal features to ground the language query into navigational actions of an agent that is navigating the video in search of the candidate clip. The agent uses a policy network trained for efficient search performance, resulting in an system that on average examines less then 50% of the video frames in order to localize a clip of interest, while achieving a high level of accuracy. We compare the performance of our methods in terms of both efficiency and accuracy with previous works that were designed to analyze 100% of the video frames to make a localization prediction. We additionally show our RL agent can be trained from the dataset training videos only and does not need access to any other data or online simulation.

CHAPTER 4

NAVIGATION AS A STRUCTURED DISTANCE PROBLEM

4.1 Introduction

In recent years, we have seen significant advances in learning-based approaches for indoor navigation [51, 58]. Impressive performance gains have been obtained for a range of tasks, from non-semantic point-goal navigation [60] to semantic tasks such as image-goal [68] and object-goal navigation [55, 71], via methods that use reinforcement learning (RL). The effectiveness of RL for these tasks can be attributed in part to the emergence of powerful new simulators such as Habitat [61], Matterport [97] and AI2Thor [98]. In Chapter 3 we have seen the extension of these RL-based navigation approaches to multi-modal navigation within the video domain. Recent advances in simulators and RL methods have led to impressive scale-ups, with RL agents learning over billions of frames from large-scale interaction data in 3D environments. But do we actually need simulation and RL to learn to navigate? Is there an alternative way to formulate the navigation problem, such that no ground-truth maps or active interaction are required? These are valuable questions to explore because learning navigation in simulation constrains the approach to a limited set of environments, since the creation of 3D assets remains costly and time-consuming.

In this chapter, we propose a self-supervised approach to learning how to navigate from passive egocentric videos. We study this approach in the context of image-goal navigation. In this task the agent is navigating to an unknown location guided by the current observations and a given image of the goal location. While there is no language component in this task, the method and findings in this chapter can be applied to language guided navigation and localization task. The contributions of this chapter is to show that navigation is a structured problem in which we can formulate a successful approach that does not need to learn

via online interaction or a simulator. The contents of this chapter appeared in [2].

Our novel method is simple and scalable (no simulator required for training), and at the same time highly effective, as it outperforms RL-based formulations by a significant margin. To introduce our approach, let us first examine the role of RL and simulation in standard navigation learning methods. In the standard RL formulation, an agent gets a reward upon reaching the goal, followed by a credit assignment stage to determine the most useful state-action pairs. But do we actually need the *reinforce* function for action credit assignment? Going a step further, do we even need to learn a policy explicitly? In navigation, we argue that the state space itself is highly structured via a distance function, and the structure itself could be leveraged for credit assignment. Simply put, states that help reduce the distance to the goal/destinations are better – and therefore distance could be used either as the value function or as a proxy for it. In fact, RL formulations frequently use 'distance reduced to goal' in reward shaping. The key property of our approach is that we learn a generalizable distance estimator *directly* from passive videos, and as a result we do not require any interaction. We demonstrate that an effective distance estimator can be learned directly from visual trajectories, without the need for an RL policy to map visual observations to the action space, thereby obviating the need for extensive interaction in a simulator and hand-designed rewards. However passive videos do not provide learning opportunities for obstacle avoidance since they rarely, if ever, consist of cameras bumping into walls. We forego the need for active interaction to reason about collisions as we show that that failed trajectories/obstacle avoidance are only required locally and simple depth maps are sufficient to prune invalid actions/location for navigation. More broadly, our approach can be considered as closely related to model-based control, which is an alternate paradigm to RL based policy learning, with the key insight that components of the model and cost functions can be learned from passive data.

No RL, No Simulator Approach (NRNS): Our NRNS algorithm can be described as follows. During training we learn two functions from passive videos: (a) a geodesic dis-

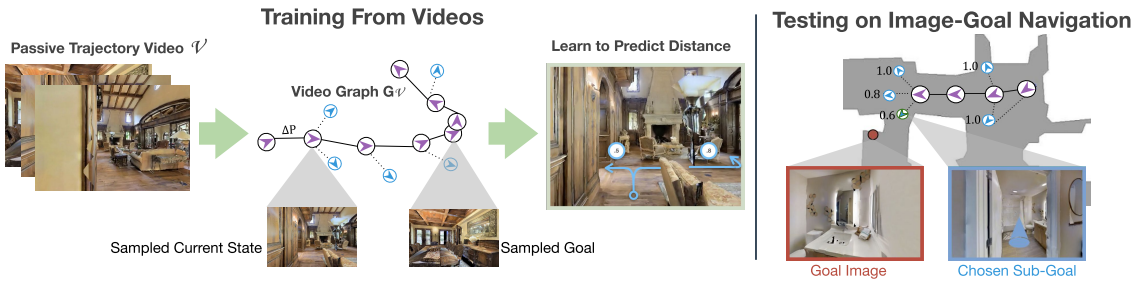

Figure 4.1: **Left**: Using passive videos we learn to predict distances for navigation. Our distance function learns the priors of the layouts of indoor buildings to estimate distances to goal location. **Right**: Image-Goal Navigation Task [70]. Our model uses distance function to predict distances of unexplored nodes and uses greedy policy to choose shortest-distance node.

tance estimator: given the state features and goal image, this function predicts the geodesic distance of the unexplored frontiers of the graph to the goal location. We then use the greedy policy where we select the node with least distance; (b) a target prediction model: Given the goal image and the image from the agent's current location, this function predicts if the goal is within sight and can be reached without collisions and the function predicts the exact location of the goal. The key is both distance model and the target prediction model can be learned from passive RGBD videos with SLAM to estimate relative poses. We believe our simple NRNS approach should act as a strong baseline for any future approaches that use RL and Simulation.

4.2 Image Goal Navigation using Topological Graphs

We propose No RL, No Simulator "NRNS", a hierarchical modular approach to image-goal navigation that comprises of: a) high-level modules for maintaining a topological map and using visual and semantic reasoning to predict sub-goals, and b) heuristic low-level modules that use depth data to select low level navigation actions to reach sub-goals and determine geometrically explorable area. We first describe NRNS in detail and later show in section 4.3 that the high-level modules can be trained without using any simulation, interaction or even ground-truth scans – that only passive video data is sufficient to learn the semantic and visual reasoning used in navigation.

4.2.1 Formulation and Representation

Task Definition. We tackle the task of image-goal navigation, where an agent is placed in a novel environment, and is tasked with navigating to an (unknown) goal position that is specified using an image taken from that position, shown in Figure 4.2. More formally, an episode begins with an agent receiving an RGB observation (I_G) corresponding to the goal position. At each time step, t, of the episode the agent receives a set of observations s_t, and must take a navigation action a_t. The agents state observations, s_t, are defined as a narrow field of view RGBD, I_t, and egocentric pose estimate, P_t. The agent must use a policy $\pi(a_t|s_t, I_G)$ to navigate to the goal before reaching a maximum number of actions.

Topological Map Representation. The NRNS agent maintains a topological map ($G(N, E)$) where a graph, defined by the nodes N and edges E, provides a sparse representation of the environment. Concretely, a node $n_i \in N$ is associated with a pose p_i, defined by location and orientation. Each node n_i can either be 'explored' *i.e.* the agent has previously visited the pose and obtained a corresponding RGBD image I_i, or 'unexplored' *e.g.* unvisited positions at the exploration frontier which may be visited in the future. Each edge $e \in E$ connects a pair of adjacent nodes n_i and n_j. Nodes are deemed adjacent only if a short and 'simple' path exists between the two nodes, as further detailed in subsection 4.2.3. Each edge between adjacent nodes is then associated with the attribute ΔP – the relative pose between the two nodes.

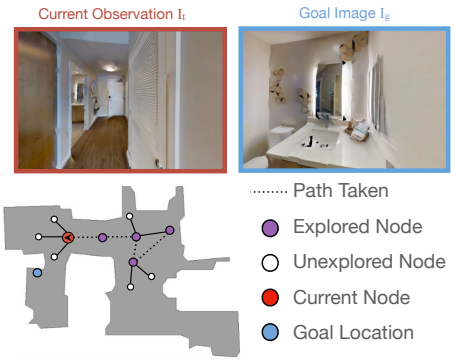

Figure 4.2: Image-Goal navigation task using a topological graph.

4.2.2 Global Policy via Distance Prediction

Given a representation of the environment as a topological graph, our global policy is tasked with identifying the next 'unexplored' node that the agent should visit, and the low-level

policy is then responsible for executing the precise actions. Intuitively, we want our agent to select the next node as the one that minimizes the total distance to goal. The global policy's inference can thus be reduced to predicting distances from nodes in our graph to the goal location. To this end, our approach leverages a distance prediction network ($\mathcal{G}_D$) which operates on top of the $G(N, E)$ to predict distance-to-goal for each unexplored node $n_{ue} \in G$. Our global policy then simply selects the node with least total distance to goal which is defined as: distance to unexplored node from the agent's current position, plus the predicted distance from unexplored node to the goal.

The input to the distance prediction network $\mathcal{G}_D$ is the current topological graph $G(N, E)_t$ and I_G. While the explored nodes have an image associated with them, the unexplored nodes naturally do not. To allow prediction in this setup, we use a Graph Convolutional Network (GCN) architecture to first induce visual features for $n_{ue} \in G$, and then predict the distance to goal using an MLP.

As illustrated in Figure 4.3, the network first encodes the RGB images at each explored node (n_i) using a ResNet18 [99] to obtain feature vector $\mathbf{h_i} \in \mathbb{R}^{512}$. Each edge $e_{i,j}$ is further represented by a feature vector $\mathbf{u_{i,j}}$, which is the flattened pose transformation matrix ($^{\mathbf{i}}\mathbf{K_j} \in \mathbb{R}^{4\times4}$). The adjacency matrix $\mathbf{A_t}$, $\mathbf{u_{i,j}}$, and $\mathbf{h_i}$ are passed through a GCN comprising of two graph attention (GAT) layers [100] with intermediate non-linearities. Note that we extend the graph attention layer architecture to additionally use edge features $\mathbf{u_{i,j}}$ when computing attention coefficients. The predicted visual features for unexplored nodes are then finally used to compute predicted distance-to-goal d_i from each node to I_G using a simple MLP.

To select the most 'promising' n_{ue} to explore, the distance from the agent's current location n_t also needs to be accounted for. For $n_{ue}, n_t \in G$, the 'travel cost' is added to d_i, calculated using shortest path on G from $n_{ue} \rightarrow n_t$. Our global policy then selects the unexplored node with the minimum total distance score as the next sub-goal.

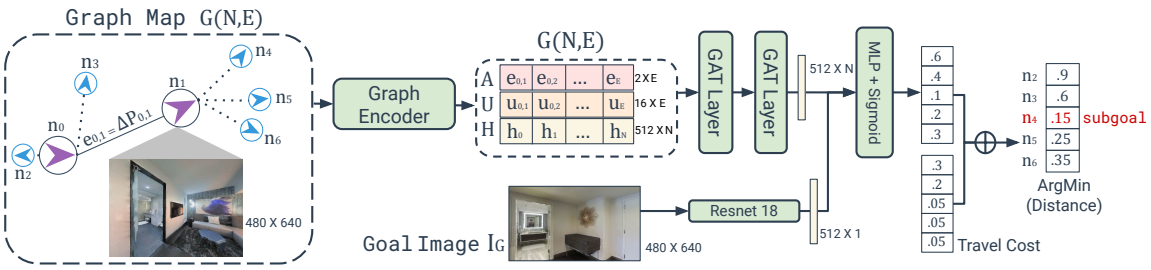

Figure 4.3: The Global Policy and $\mathcal{G}_D$ architecture used to model distance-to-goal prediction. $\mathcal{G}_D$ employs a Resnet18 encoder, Graph Attention layers and multi-layer perception with sigmoid.

4.2.3 Local Navigation and Graph Expansion

The NRNS global policy selects the sub-goal that the agent should pursue, and the precise low-level actions to reach the sub-goal are executed by a heuristic local policy. After the local policy finishes execution, the NRNS agent updates the graph to include the new observations and expands the graph with unexplored nodes at the exploration frontier.

Local Policy. The NRNS local policy, denoted as $\mathcal{G}_{LP}$, receives a target position, defined by distance and angle (ρ_i, ϕ_i) with respect to the agent's current location. When $\mathcal{G}_D$ outputs a sub-goal node, ρ_i, ϕ_i are calculated from the current position and passed to $\mathcal{G}_{LP}$. Low level navigation actions are selected and executed using a simplistic point navigation model based on the agents egocentric RGBD observations and (noisy) pose estimation. To navigate towards its sub-goal, the agent builds and maintains a local metric map using the noisy pose estimator and depth input. This effectively allows it to reach local goals and avoid obstacles. The local metric maps are discarded upon reaching the sub-goal, as they are based on a noisy pose estimator. This policy is adapted from [70] and is also used for Image-Goal Navigation in [68].

Explorable Area Prediction for Graph Expansion. We incorporate a 'graph expansion' step after the agent reaches a sub-goal via $\mathcal{G}_{LP}$ and before the agent selects a new sub-goal. First, the agent updates $G(N, E)$ to record the current location as an explored node and store the associated RGBD observation. Second, the agent creates additional 'unexplored'

nodes, n_{ue}, adjacent to the current node, n_t based on whether the corresponding directions are deemed 'explorable'. We use a explorable area prediction module, $\mathcal{G}_{EA}$, to determine which adjacent areas to the current location are geometrically explorable. This is untrained, heuristic function takes the egocentric depth image $I_{t_{depth}}$ and tests 9 angles in the direction θ from the current position and returns the angles θ that are not blocked by obstacles within a depth of 3 meters. The NRNS agent tests θ at [0,$\pm$15,$\pm$30,$\pm$45,$\pm$60], these angles are chosen based on the agent's turn radius of 15°and 120°FOV. For all θ determined to be explorable, the agent updates $G(N, E)$ by adding an 'unexplored' node at position $\rho = 1m$ and $\phi = \theta$ relative to the agent in, with the corresponding edge. If a node already exists in one of the explorable areas at a similar position, only an edge is added to $G(N, E)$ and not a new node.

4.2.4 Putting it Together

Stopping Criterion. The above described modules ($\mathcal{G}_D$, $\mathcal{G}_{LP}$, $\mathcal{G}_{EA}$) allow the NRNS agent to continue intelligently exploring to reach the target image. To allow the agent to finish navigating when it is near the goal, we additionally learn a target prediction network $\mathcal{G}_T$ that serves as a stopping criterion. Given I_G and current image I_t, $\mathcal{G}_T$ is simple MLP that predicts: a) a probability $\beta_s \in [0, 1]$ indicating whether the goal is within sight, and if so, b) the relative position (distance, direction) of the goal ρ_g, ϕ_g.

Algorithm. We outline our navigation algorithm in Figure 4.4 and also describe it in detail below. An example result on the image-goal navigation task is shown in Figure 4.6. Among the modules used, the local policy $\mathcal{G}_{LP}$ and explorable area $\mathcal{G}_{EA}$ modules do not require any learning. As we show in section 4.3, $\mathcal{G}_D$ and $\mathcal{G}_T$ can both be learned using only passive data.

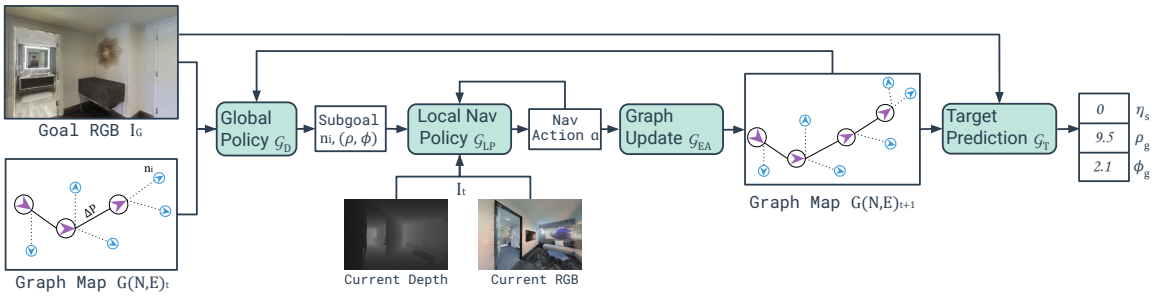

Figure 4.4: The hierarchical modular NRNS approach to the Image-Goal Navigation task, for a single time step in the episode. The global policy $\mathcal{G}_D$ selects an unexplored node n_i as a sub-goal. The sub-goal position (ρ_i, ϕ_i) is passed to the local navigation policy $\mathcal{G}_{LP}$ which takes in the current RGBD observations and outputs low level actions until the agent reaches n_i. The graph is then updated with the current observations I_{t+1} and new unexplored nodes and edges generated by $\mathcal{G}_{EA}$.

Algorithm 1: NRNS Image Navigation

```
// initialize graph
n_0 = (P_{t=0}, I_{t=0}); e_0 = (n_0, n_0);
N = (n_0); E = (e_0);
G = (N, E);
// loop until reached goal or max steps
while steps taken ¡ max steps do
    n_{i+1}, ..., n_{i+k} = G_EA(I_t);  // determine valid explorable areas
    N += n_{i+1}, ..., n_{i+k}; E += e_{t,i+1}, ..., e_{t,i+k};  // add unexplored nodes & edges
    n_sg, (ρ_sg, φ_sg) = argmin(G_D(G, I_G) + TravelCost(G, n_t));  // select sub-goal
    I_{t+1}, P_{t+1} = G_LP(ρ_sg, φ_sg);  // navigate to sub-goal
    n_sg = (P_{t+1}, I_{t+1}) // update graph with observations
    β_s, (ρ_g, φ_g) = G_T(I_{t+1}, I_G);  // stopping criterion
    if β_s ¿ .5 then
        G_LP(ρ_g, φ_g);  // navigate to target
        break;
    end
end
```

4.3 Learning from Passive Data

The learned high-level policies of NRNS are the Distance Network $\mathcal{G}_D$ and Target Prediction Network $\mathcal{G}_T$. A key contribution of our work is showing that these functions can be learned from passive data alone. This eliminates the need for online interaction and ground-truth maps, which allows us to train the NRNS algorithm without using RL or any

simulation.

Learning Distance Prediction. First, we describe how to learn the function $\mathcal{G}_D$. Given a topological graph (consisting of both explored and unexplored nodes) and a goal image as input, $\mathcal{G}_D$ predicts the geodesic distance between all unexplored nodes and the goal image. Our training data therefore is of triplet form: (G, I_G, D_U) where G is a topological graph, I_G is a goal image, and D_U is the ground-truth distance of unexplored nodes to the goal location (we use L2-Loss on distance function).

We sample training graphs using passive videos in a two-step process. In the first step, we convert the whole video to a long topological graph. The graph contains both explored and unexplored nodes. We approximate distance to unexplored nodes using the geodesic distance along the trajectory. In the second step, we uniformly sample sub-graphs and goal locations over each video's topological graph.

Step 1: Video to Topological Graph. In order to train with passive video data, we create a topological graph, $G_{\mathcal{V}_i}$, for every video $\mathcal{V}_i$ in the passive dataset. First, we generate a stepwise trajectory based on odometry information from each frame. We adapt this trajectory to topological graph structure using affinity clustering [101] on the stepwise graph via visual features and odometry information. Visual features for each frame are extracted via a Places365 [102] pretrained Resnet18. Each cluster represents a single node in the topological graph $G_{\mathcal{V}_i}$ and stepwise visual features of frames in the cluster are average pooled. The topological graphs are then expanded to have unexplored nodes. These are created using $\mathcal{G}_{EA}$ over the RGBD of each node centroid in $G_{\mathcal{V}_i}$. Figure 4.5 shows an example video and transformation between the stepwise trajectory and the topological graph.

Step 2: Sampling training datapoints. Individual data instances are selected via uniform sampling without replacement. This means selecting a random node in $G_{\mathcal{V}}$ as the goal location and a random sub-graph of $G_{\mathcal{V}}$ as the observed trajectory and current location. Distance along the trajectory is used as the ground truth distance label between nodes in the sub-graph and the goal image: these distance labels are used to train the network $\mathcal{G}_D$.

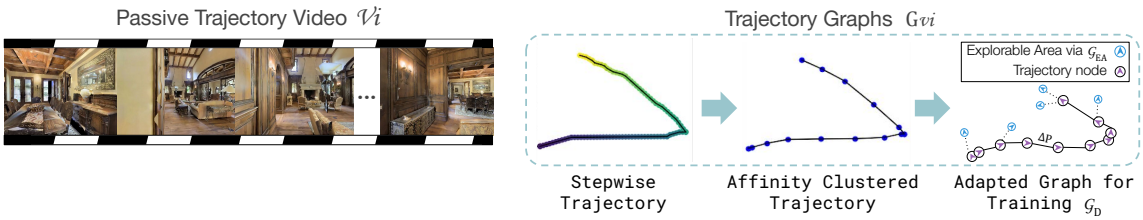

Figure 4.5: Example from the passive video dataset. Frames of a video trajectory $\mathcal{V}_i$ are shown on the left. The stepwise trajectory is then turned into a trajectory graph $G_{\mathcal{V}_i}$ via affinity clustering [101] of node image and pose features. $G_{\mathcal{V}_i}$ is adapted to train the Global Policy $\mathcal{G}_D$ and target direction prediction $\mathcal{G}_E A$. An example of an adapted $G_{\mathcal{V}_i}$ for training is shown on the left.

It should be noted that the distance labels are partially noisy due to non-optimal long term paths of the video trajectories, making this a challenging modeling problem. In total $\mathcal{G}_D$ is trained over 75k training instance for Gibson and 148k instances for MP3D.

Learning Target Direction Prediction The $\mathcal{G}_T$ prediction network receives a goal image and the current node image as input, and predicts whether the goal is within sight of current image. To gather training instances for $\mathcal{G}_T$, we make a simplifying assumption that any adjacent pair of nodes (similar features and odometry) in the topological graph are positive examples and any other pair of nodes are negative examples.

4.4 Experimental Setup

Image-Goal Navigation Task Setup. At the beginning of each episode, the agent is placed in an unseen environment. The agent receives observations from the current state, a 3 x 1 odometry pose reading and RGBD image, and the RGB goal image. Observation and goal images are 120°FOV and of size 480 x 640. An episode is considered a success if the agent is able to reach within $1m$ of the goal location and do so within a maximum episode length of 500 steps. Each episode is also evaluated by the efficiency of the navigational path from start to goal, which is quantitatively measured by Success weighted by inverse Path Length (SPL) [51]. Note, the narrow field of the agent in this task definition differs from past works which use panoramic views [68]. The decision to use a narrow field is based

on our method of training only on passive data. Current passive video datasets of indoor trajectories such as YouTube Tours [69], RealEstate10k [103] and our NRNS dataset, do not contain panoramas.

Action Space. The agent's action space contains four actions: `forward` by .25m, `rotate left` by 15°, `rotate right` by 15°, and `stop`. In our experiments, we consider two cases for pose estimation and action transition. In the first condition the agent has access to ground truth pose and the navigation actions are deterministic. In the second condition, noise is added to the pose estimation and the actuation of the agent. We utilize the realistic pose and actuation noise models from [70], which are similarly used in [68]. The actuation noise adds stochastic rotational and translations transitions to the agent's navigational actions.

Training Data. Our key contribution is the ability to learn navigation agents from passive data. In theory, our approach can be trained from any passive data source, and we test this in Sec. subsection 4.4.2 using RealEstate10K [103]. However, since RL-based baselines are trained in the Habitat Simulator [61], we generate our passive video dataset, of egocentric trajectory videos, using the same training scenes to provide direct comparison and isolate domain gap issues.

Specifically, the trajectory videos are created as follows. A set of 2 - 4 points are randomly selected from the environment using uniform sampling. A video is then generated of the concatenated RGBD frames of the shortest path between consecutive points. Note that the complete video trajectory is not step-wise optimal nor is the end frame of the trajectory. Frames in the videos are of size 480 X 640 and have a FOV 120°and each frame is associated with a 3 x 1 odometry pose reading. In the noisy setting discussed in section 4.4, sensor and actuation noise is injected into training trajectories. We create 19K, 43K video trajectories, containing 1, 2.5 million frames respectively, on the Gibson and MP3D datasets. We then use this data to train the NRNS modules, as described in section 4.3.

Test Environments. We evaluate our approach on the task of image-goal navigation. For

testing, we use the Habitat Simulator [61]. We evaluate on the standard test-split for both the Gibson [104] and Matterport3D (MP3D) [18] datasets. For MP3D, we evaluate on 18 environments and for Gibson, we evaluate on 14 environments.

Baselines. We consider a number of baselines to contextualize our Image-Goal Navigation results:

- **BC w/ ResNet + GRU.** Behavioral Cloning (BC) policy where I_t and I_G are encoded using a pretrained ResNet-18. Both image encodings and the previous action a_{t-1} are passed through a two layer Gated Recurrent Unit (GRU) with softmax, which outputs the next action a_t.
- **BC w/ ResNet + Metric Map.** BC policy: I_t and I_G are encoded with a ResNet, same as the above policy. This policy keeps a metric map built from the depth images. The metric map is encoded with a linear layer. The metric map encoding and encodings of I_t and I_G are concatenated and passed into an MLP with softmax, which outputs the next navigational action a_t.
- **End to End RL with DDPPO.** An agent is trained end to end with proximal policy optimization [54] in the Habitat Simulator [61] for the image-goal navigation task.

We use and adapt code for baseline algorithms from Chaplot, Salakhutdinov, Gupta, and Gupta [68]. However, as our setup uses narrow-view cameras instead of panoramas, these adapted baselines perform worse compared to their previously reported performance. This difference in setup also makes direct comparison with [68] infeasible, as they critically rely on panoramic views for localization. The baseline behavioral cloning policies are also trained only using the passive dataset described in section 4.3. This makes the BC baseline policies directly comparable to the NRNS model. The end to end RL policy is trained with DDPPO [54] for 20 million steps for each dataset, respectively. Training the $\mathcal{G}_D$ model takes 20-25 epochs, requiring $\sim$6 hours on a single GPU. Training the $\mathcal{G}_T$ model takes 10-15 epochs, requiring $\sim$2 hours on a single GPU.

Episode Settings. To provide an in-depth understanding of the successes and limitations

of our approach, we sub-divide test episodes into two categories: 'straight' and 'curved'. In 'straight' episodes, the ratio of shortest path geodesic-distance to euclidean-distance between the start and goal locations is < 1.2 and rotational difference between the orientation of the start position and goal image is $< 45°$. All other start-goal location pairs are labeled as 'curved' episodes. We make this distinction due to the nature of the narrow field of view of our agent, which strongly effects performance on curved episodes, since the agent must learn to turn both as part of navigating and part of seeking new information about the target location. Also, while a greedy policy being successful on 'straight' episodes might be expected, a competitive performance on even 'curved' episodes will highlight how effective our simple model and policy is. We further subdivide each of these 2 categories into 3 difficulty sub-categories: 'easy', 'medium' and 'hard'. Difficulty is determined by length of the shortest path between the start and goal locations. Following [68], the 'easy', 'medium' and 'hard' settings are $(1.5 - 3m)$, $(3 - 5m)$, and $(5 - 10m)$ respectively. To generate test episodes we uniformly sample the test scene for start-goal location pairs, to create approximately 1000 episodes per setting.

4.4.1 Results

Tables Table 4.1, Table 4.2 show the performance of our NRNS model and relevant baselines on the test splits of the Gibson and Matterport datasets. In Figure 4.6, we visualize an episode of the NRNS agent as it navigates to the goal-image.

NRNS outperforms baselines. Our NRNS algorithm significantly outperforms the BC and end to end RL policies in terms of Success and SPL @ 1m on both datasets – improving upon the best baseline, a Behavioral Cloning (BC) policy with a ResNet and GRU, across splits of Gibson by an absolute 20+% on Straight episodes and 10+% on Curved episodes. We find that a BC policy using a GRU for memory outperforms using only a metric map. We attribute this to a spatial memory (metric map) being less informative for agent exploration than a memory of the visual features and previous steps (GRU). We observe that an

Figure 4.6: Example of an Image-Goal Navigation episode on MP3D. Shows the agent's observations and internal topological graph at different time steps.

end to end RL policy trained in simulation performs much weaker than all baselines. The poor performance of target-driven RL methods for image-goal navigation is unsurprising [68, 57]. This demonstrates the difficulty of learning rewards on low level actions instead of value learning on possible exploration directions, exacerbating the difficulty of exploration in image-goal navigation. Adding to the challenges of the task, all policies must learn the stop action. Previous works [68] have found that adding oracle stopping, to a target-driven RL agent, leads to large gains in performance on image-goal navigation. The limitations of all approaches are seen on the 'hard' and 'curved' episode settings, showing the overall difficulty of the exploration problem and the challenge of using a narrow field of view.

NRNS is robust to noise. Even with the injection of sensor and actuation noise [70], in both the passive training data and test episodes, NRNS maintains superior performance to all baselines. In fact, we find that the addition of noise leads to only an absolute drop in success between .8-8% on Gibson [104] and 1-5% on MP3D [18]. An interesting observation is small increase in performance (w/noise) for the hard-case. We believe this is because gt-distance for hard cases are more error prone and noise during training provides regularization.

Total

NRNS Module Ablations. Table 4.3 and Table 4.4 report detailed ablations of NRNS. We ablate the NRNS approach by testing each module individually. In the ablation exper-

Table 4.1: Comparison of our model (NRNS) with baselines on Image-Goal Navigation on **Gibson**[104]. We report average Success and Success weighted by inverse Path Length (SPL) @ $1m$. Noise refers to injection of sensor & actuation noise into the train videos and test episodes. * denotes using simulator.

		Easy		Medium		Hard	
Path Type	Model	Succ ↑	SPL ↑	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	End-to-End RL * [54]	3.60	2.73	5.80	5.01	3.47	3.28
	BC w/ ResNet + Metric Map	24.80	23.94	11.50	11.28	1.36	1.26
	BC w/ ResNet + GRU	34.90	33.43	17.60	17.05	6.08	5.93
	NRNS w/ noise	64.10	55.43	47.90	39.54	**25.19**	18.09
	NRNS w/out noise	**68.00**	**61.62**	**49.10**	**44.56**	23.82	**18.28**
Curved	End-to-End RL* [54]	4.50	2.93	4.40	3.81	1.50	1.43
	BC w/ ResNet + Metric Map	3.10	2.53	0.80	0.71	0.20	0.16
	BC w/ ResNet + GRU	3.60	2.86	1.10	0.91	0.50	0.36
	NRNS w/ noise	27.30	10.55	23.10	10.35	10.50	5.61
	NRNS w/out noise	**35.50**	**18.38**	**23.90**	**12.08**	**12.50**	**6.84**

Table 4.2: Comparison of our model (NRNS) with baselines on Image-Goal Navigation on **MP3D**[18].

		Easy		Medium		Hard	
Path Type	Model	Succ ↑	SPL ↑	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	End-to-End RL* [54]	7.50	4.00	3.50	1.73	1.00	0.55
	BC w/ ResNet + Metric Map	25.80	24.82	11.30	10.65	3.00	2.93
	BC w/ ResNet + GRU	30.20	29.57	12.70	12.48	4.40	4.34
	NRNS w/ noise	63.80	53.12	36.20	26.92	**24.10**	16.93
	NRNS w/out noise	**64.70**	**58.23**	**39.70**	**32.74**	22.30	**17.33**
Curved	End-to-End RL* [54]	4.90	1.78	3.20	1.37	1.10	0.46
	BC w/ ResNet + Metric Map	4.90	4.23	1.40	1.29	0.40	0.34
	BC w/ ResNet + GRU	3.10	2.61	0.80	0.77	0.10	0.02
	NRNS w/ noise	21.40	8.19	15.40	6.83	**10.0**	4.86
	NRNS w/out noise	**23.70**	**12.68**	**16.20**	**8.34**	9.10	**5.14**

iments, we replace the module output with the ground truth labels or numbers in order to evaluate the affect of each module on the performance of the overall approach. For simplicity, all ablations are trained and tested without sensor or actuation noise. Unsurprisingly, we find that the Global Policy, $\mathcal{G}_D$, has a large affect on performance (Row 4 and 8). We find that the largest affects are seen in the 'hard' and 'curved' test episodes. This is unsurprising because as the distance to the goal increases and the path increases in complexity and the search space of $\mathcal{G}_D$ increases.

Table 4.3: Ablations of NRNS with baselines on Image-Goal Navigation on **Gibson** [104]. We report average Success and Success weighted by inverse Path Length (SPL) @ $1m$. ✗ denotes a module being replaced by the ground truth labels and a ✓ denotes the NRNS module being used.

	NRNS Ablation			Easy		Medium		Hard	
Path Type	$\mathcal{G}_{EA}$	$\mathcal{G}_T$	$\mathcal{G}_D$	Succ ↑	SPL ↑	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	✗	✗	✗	100.00	99.75	100.00	99.62	100.00	99.57
	✓	✗	✗	99.90	99.05	98.20	95.06	95.91	90.53
	✓	✓	✗	79.40	73.48	71.30	67.48	62.16	58.06
	✓	✓	✓	68.00	61.62	49.10	44.56	23.45	18.84
Curved	✗	✗	✗	100.00	97.62	100.00	97.47	100.00	98.18
	✓	✗	✗	99.70	95.93	97.50	90.14	89.30	79.95
	✓	✓	✗	65.00	56.70	58.10	52.51	47.70	42.28
	✓	✓	✓	35.50	18.38	23.90	12.08	12.50	6.84

4.4.2 Training on Passive Videos in Wild

Finally, we demonstrate that our model can be learned from passive videos in the wild. Towards this end, we train our NRNS model using the RealEstate10K dataset [103] which contains YouTube videos of real estate tours. This dataset has 80K clips with poses estimated via SLAM. Note that the average trajectory length is smaller than test time trajectories in Gibson or MP3D, and we therefore only evaluate on 'easy' and 'medium' settings. Table 4.5 shows the performance. Few things to note: there is drop in performance as compared to training using Gibson videos, which we attribute to domain shift. Even after this drop, our approach, trained on real-world passive videos, outperforms several baselines which are trained and tested on Gibson itself. We find these results to be a strong indication of the effectiveness of our algorithm.

RealEstate10k Dataset Description. RealEstate10K [103] is a large video dataset of trajectories through mostly indoor scenes. 80k video clips, containing $\sim$10 million frames each corresponding to a provided camera pose. The poses are procured from SLAM and bundle adjustment algorithms run on the videos, and they represent the orientation and path of the camera along the trajectory. The clips are gathered from 10k YouTube videos of real estate footage. The clips are relatively short and range between 1-10 seconds [103]. While

Figure 4.7: Comparison of the passive videos from different datasets used for training our NRNS agent. MP3D and Gibson passive video frames are images of rendered environments using the habitat simulator and therefore are similar in photo realism. RealEstate10K video frames are taken directly from a real estate tour YouTube video and therefore differ from MP3D and Gibson.

Table 4.4: Ablations of NRNS with baselines on Image-Goal Navigation on **MP3D** [18]. We report average Success and Success weighted by inverse Path Length (SPL) @ $1m$. ✗ denotes a module being replaced by the ground truth labels and a ✓ denotes the NRNS module being used.

	NRNS Ablation			Easy		Medium		Hard	
Path Type	$\mathcal{G}_{EA}$	$\mathcal{G}_T$	$\mathcal{G}_D$	Succ ↑	SPL ↑	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	✗	✗	✗	100.00	100.00	99.80	99.07	100.00	99.02
	✓	✗	✗	100.00	100.00	99.00	96.81	96.10	92.65
	✓	✓	✗	74.20	68.19	64.50	60.13	58.40	55.38
	✓	✓	✓	64.70	58.23	39.70	32.74	22.30	17.33
Curved	✗	✗	✗	100.00	94.08	99.90	95.39	100.00	97.00
	✓	✗	✗	100.00	93.06	97.70	90.22	91.60	0.82.87
	✓	✓	✗	62.20	53.31	54.10	47.51	51.00	44.92
	✓	✓	✓	23.70	12.68	16.20	8.34	9.10	5.14

the total number of frames in the RealEstate10k clips is large, the total length of the trajectory in meters is on average shorter than the MP3D and Gibson videos. Figure Figure 4.7 shows the visual difference between frames of the passive video dataset created from the simulator and those taken from YouTube videos.

Table 4.5: Comparison of our model (NRNS) trained with different sets of passive video data, on Image-Goal Navigation on **Gibson** [104]. We report average Success and SPL @ $1m$. Results shown are tested without sensor & actuation noise.

			Easy		Medium	
Path Type	Training Data	Model	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	RealEstate10k [103]	NRNS	56.42	48.01	30.30	25.67
	MP3D	NRNS	59.80	52.35	37.00	31.89
	Gibson	NRNS	68.00	61.62	49.10	44.56
	Gibson	BC w/ ResNet + GRU	30.20	29.57	12.70	12.48
Curved	RealEstate10k [103]	NRNS	21.10	15.76	12.90	5.57
	MP3D	NRNS	28.26	13.59	11.00	5.10
	Gibson	NRNS	35.50	18.38	23.90	12.08
	Gibson	BC w/ ResNet + GRU	3.10	2.61	0.80	0.77

4.5 Conclusion

As simulators become faster, they maintain the problem of being limited to a small number of train environments that are expensive to generate. In contrast, large numbers of passive

Table 4.6: Comparison of our model (NRNS) trained with different sets of passive video data, on Image-Goal Navigation on **MP3D** [18]. We report average Success and SPL @ $1m$. Results shown are tested without sensor & actuation noise.

			Easy		Medium	
Path Type	Training Data	Model	Succ ↑	SPL ↑	Succ ↑	SPL ↑
Straight	RealEstate10k [103]	NRNS	44.58	39.27	15.81	10.73
	Gibson	NRNS	59.20	54.12	22.90	19.34
	MP3D	NRNS	64.70	58.23	39.70	32.74
	MP3D	BC w/ ResNet + GRU	30.20	29.57	12.70	12.48
Curved	RealEstate10k [103]	NRNS	9.43	4.96	5.30	2.86
	Gibson	NRNS	12.10	5.66	8.48	4.92
	MP3D	NRNS	23.70	12.68	16.20	8.34
	MP3D	BC w/ ResNet + GRU	3.10	2.61	0.80	0.77

videos of diverse sets of environments can be inexpensively gathered from the internet or captured. Our presented approach, NRNS, neither requires access to ground-truth maps nor online policy interaction. We study this alternate approach in the context of Image-Goal Navigation and demonstrate this approach can outperform end-to-end RL methods and behavioral cloning policies. We believe that this approach can be successfully extended to embodied navigation tasks which contain natural language components. We believe it would be possible to structure the video navigation via activity description task, discussed in chapter 3, in a similar modular approach to NRNS for image-goal navigation but with the addition of multi-modal representations for explored and unexplored video regions.

CHAPTER 5
LANGUAGE GUIDED LOCALIZATION OF EMBODIED AGENTS

In this chapter we expand upon the localization and navigation tasks introduced in chapters 3 and 4, respectively, and develop a set of novel language-guided localization tasks for embodied agents. These tasks provide a challenging multi-agent setting for exploring learning approaches and representations.

5.1 Introduction

Automated interaction with humans in everyday activity remains a significant challenge for artificial intelligence (AI). Current interactive systems take many forms and operate on different time scales; examples include shopping recommendations, conversational AI, autonomous vehicles, and social robots. However, these models typically only work within a narrow range of conditions. A significant challenge is presented by the prospect of all-day wearable augmented reality (AR) glasses or a robotic home-aid: the ideal is an automated system that that can offer assistance in any context. The AI required for such a system would likely require "theory of mind" similar to that of humans, whereby people infer goals and cognitive states of others and take strategic, context-dependent actions that may be cooperative or adversarial in nature. To make progress toward this system it is necessary to develop new embodied tasks which require communication and collaboration.

Imagine getting lost in a new building on your way to a meeting. Unsure of exactly where you are, you call your friend and start describing your surroundings (*'I'm standing near a big blue couch in what looks like a lounge. There are a set of wooden double doors opposite the entrance.'*) and navigating in response to their questions (*'If you go through those doors, are you in a hallway with a workout room to the right?'*). After a few rounds of dialog, your friend who is familiar with the building will hopefully know your location.

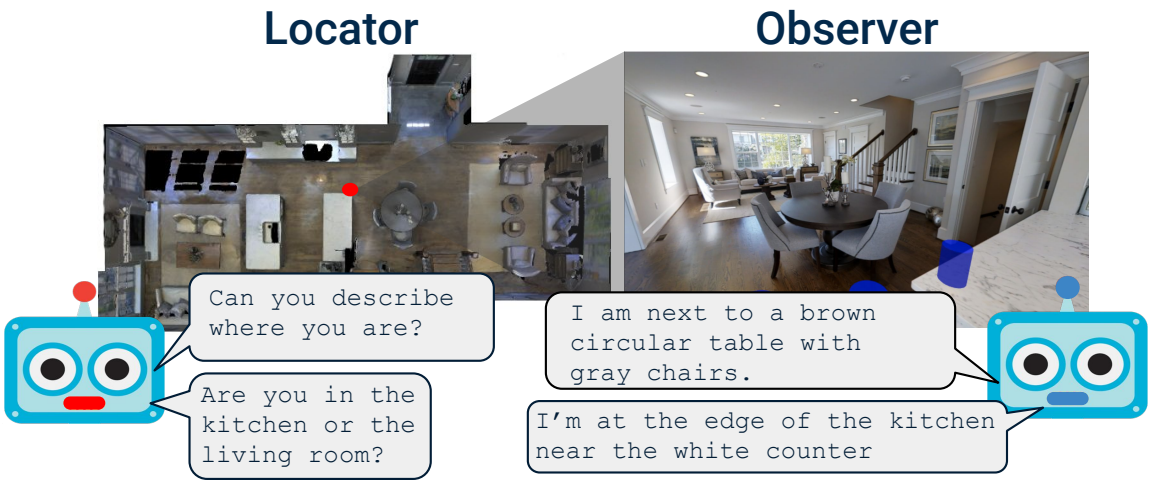

Figure 5.1: LED Task: The Locator has a top-down map of the building and is trying to localize the Observer by asking questions and giving instructions. The Observer has a first person view and may navigate while responding to the Locator. The turn-taking dialog ends when the Locator predicts the Observer's position.

Looking to the future we can clearly imagine a scenario where this friend is replaced with a personal AI assistant, or that you are communicating with a personal robot that is trying to get to you in a new building for which it does not have a map. However success at this cooperative task involves overcoming significant challenges. The task requires goal-driven questioning based on the locator's understanding of the environment, unambiguous answers communicating observations via language, and active perception and navigation to investigate the environment and seek out discriminative observations.

In this thesis we present WHERE ARE YOU? (WAY), a new dataset based on this scenario. The contents of this chapter appeared in [3]. As shown in Figure 5.1, during data collection we pair two annotators: an Observer who is spawned at random in a novel environment, and a Locator who must precisely localize the Observer in a provided top-down map. The map can be seen as a proxy for familiarity with the environment – it is highly detailed, often including multiple floors, but does not show the Observer's current or initial location. In contrast to the "remote" Locator, the Observer navigates within the environment from a first-person view but without access to the map. To resolve this information asymmetry and complete the task, the Observer and the Locator communicate in a

live two-person chat. The task concludes when the Locator makes a prediction about the current location of the Observer. For the environments we use the Matterport3D dataset [18] of 90 reconstructed indoor environments. In total, we collect ∼6K English dialogs of humans completing this task from over 2K unique starting locations.

The combination of localization, navigation, and dialog in WAY provides for a variety of modeling possibilities. We identify three compelling tasks encapsulating significant research challenges:

– Localization from Embodied Dialog. LED, which is the main focus of this chapter, is the state estimation problem of localizing the Observer given a map and a partial or complete dialog between the Locator and the Observer. Although localization from dialog has not been widely studied, we note that indoor localization plays a critical role during calls to emergency services [105]. As 3D models and detailed maps of indoor spaces become increasingly available through indoor scanners [18], LED models could have the potential to help emergency responders localize emergency callers more quickly by identifying locations in a building that match the caller's description.

– Embodied Visual Dialog. EVD is the navigation and language generation task of fulfilling the Observer role. This involves using actions and language to respond to questions such as *'If you walk out of the bedroom is there a kitchen on your left?'* In future work we hope to encourage the transfer of existing image-based conversational agents [6] to more complex 3D environments additionally requiring navigation and active vision, in a step closer to physical robotics. The WAY dataset provides a testbed for this.

– Cooperative Localization. In the CL task, both the Observer and the Locator are modeled agents. Recent position papers [106, 107, 108] have called for a closer connection between language models and the physical world. However, most reinforcement learning for dialog systems is still text-based [109] or restricted to static images [19, 20]. Here, we provide a dataset to warm-start and evaluate goal-driven dialog in a realistic embodied setting.

Our main modeling contribution is a strong baseline model for the LED task based on LingUnet [24]. In previously unseen test environments, our model successfully predicts the Locator's location within 3 meters 32.7% of the time, vs. 70.4% for the human Locators using the same map input, with random chance accuracy at 6.6%. We include detailed studies highlighting the importance of data augmentation and residual connections. Additionally, we characterize the biases of the dataset via unimodal (dialog-only, map-only) baselines and experiments with shuffled and ablated dialog inputs, finding limited potential for models to exploit unimodal priors.

Contributions: To summarize:

1. We present WAY, a dataset of $\sim$6k dialogs in which two humans with asymmetric information complete a cooperative localization task in reconstructed 3D buildings.
2. We define three challenging tasks: Localization from Embodied Dialog (LED), Embodied Visual Dialog, and Cooperative Localization.
3. Focusing on LED, we present a strong baseline model with detailed ablations characterizing both modeling choices and dataset biases.

5.2 Comparison of Language in the WAY Dataset

A number of 'Embodied AI' tasks combining language, visual perception, and navigation in realistic 3D environments have recently gained prominence, however these tasks utilize only a single question or instruction input. Recently several papers have extended the VLN task [97] to dialog settings. The closest vision-dialog dataset to ours is Cooperative Vision-and-Dialog Navigation (CVDN) [5]. CVDN is a dataset of dialogs in which a human assistant with access to visual observations from an oracle planner helps another human complete a navigation task. CVDN dialogs are set in the same Matterport3D buildings [18] and like ours they are goal-oriented and easily evaluated. The main difference is that we focus on localization rather than navigation. Qualitatively, this encourages more descriptive utterances from the first-person agent (rather than eliciting short questions).

Table 5.1: Comparison of the language between the WAY dataset and related embodied perception datasets.

Method	Dataset Size	Vocab Size	Avg Text Length	Noun Density	Adj Density	Preposition Density	Dialog
CVDN	2050	2165	52	0.20	0.06	0.09	Yes
TtW	10K	7846	110	0.20	0.07	0.11	Yes
VLN	21K	3459	29	0.27	0.03	0.17	No
WAY	6154	5193	61	0.30	0.12	0.18	Yes

The WAY dataset tasks are also related to Talk the Walk [110] which presented a dataset for a similar task in an outdoor setting using a restricted, highly-abstracted map which encouraged language that is grounded in the semantics of building types rather than visual descriptions of the environment.

To fully illustrate the richness of the dialog contained in the WAY dataset, Table 5.1 compares the language in WAY against existing embodied perception datasets. Specifically, size, length and the density of different parts of speech (POS) are shown. Vocab size was determined by the total number of unique words. We used the [111] POS tagger to calculate the POS densities over the text in each dataset. We find that WAY has a higher density of adjectives, nouns, and prepositions than related datasets suggesting the dialog is more descriptive than the text in existing datasets.

5.3 Where Are You? (WAY) Dataset

We present the WHERE ARE YOU? (WAY) dataset consisting of 6,134 human embodied localization dialogs across 87 unique indoor environments.

Environments. We build WAY on Matterport3D [18], which contains 90 buildings captured in 10,800 panoramic images. Each building is also provided as a reconstructed 3D textured mesh. This dataset provides high-fidelity visual environments in diverse settings including offices, homes, and museums – offering numerous objects to reference in localization dialogs. We use the Matterport3D simulator [4] to enable first-person navigation between panoramas.

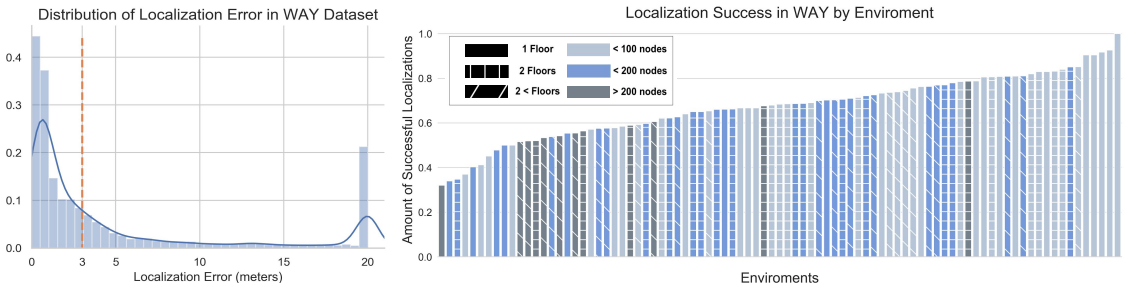

Figure 5.2: Left: Distribution of human localization error in WAY (20+ includes wrong floor predictions). Right: Human success rates (error <3m) by environment. Bar color indicates environment size (number of nodes) and pattern the number of floors.

Task. A WAY episode is defined by a starting location (i.e. a panorama p) in an environment e. The Observer is spawned at p_0 in e and the Locator is provided a top-down map of e (see Figure 5.1). Starting with the Locator, the two engage in a turn-based dialog $(L_0, O_0, \ldots L_{T-1}, O_{T-1})$ where each can pass one message per turn. The Observer may move around in the environment during their turn, resulting in a trajectory $(p_0, p_1, \ldots, p_T)$ over the dialog. The Locator is not embodied and does not move but can look at the different floors of the house at multiple angles. The dialog continues until the Locator uses their turn to make a prediction $(\hat{p}_T)$ of the Observer's current location (p_T). The episode is successful if the prediction is within k meters of the true *final* position – i.e. $||p_T - \hat{p}_T||_2 < k$ m. This does not depend on the initial position, encouraging movement to easily-discriminable locations.

Map Representation. The Locator is shown top-down views of Matterport textured meshes as environment maps. In order to increase the visibility of walls in the map (which may be mentioned by the Observer), we render views using perspective rather than orthographic projections (see left in Figure 5.1). We set the camera near and far clipping planes to render single floors such that multi-story buildings contain an image for each floor.

5.3.1 Collecting Human Localization Dialogs

To provide a human-performance baseline and gather training data for agents, we collect human localization dialogs in these environments.

Episodes. We generate 2020 episodes across 87 environments by rejection sampling to avoid spatial redundancy. For each environment, we iteratively sample start locations, rejecting ones that are within 5m of already-sampled positions. Three environments were excluded due to their size (too large or small) or poor reconstruction quality.

Data Collection. We collect dialogs on Amazon Mechanical Turk (AMT) – randomly pairing workers into Observer or Locator roles for each episode. The Observer interface includes a first-person view of the environment and workers can pan/tilt the camera in the current position or click to navigate to adjacent panoramas. The Locator interface shows the top-down map of the building, which can be zoomed and tilted to better display the walls. Views for each floor can be selected for multi-story environments. Both interfaces include a chat window where workers can send their message and end their dialog turn. The Locator interface also includes the option to make their prediction by clicking a spot on the top-down map – terminating the dialog. Note this option is only available after two rounds of dialog. Refer to the appendix for further details on the AMT interfaces.

Before starting, workers were given written instructions and a walk-through video on how to perform their role. We restricted access to US workers with at least a 98% success rate over 5,000 previous tasks. Further, we restrict workers from repeating tasks on the same building floor. In order to filter bad-actors, we monitored worker performance based on a running-average of localization error in meters and the number of times they disconnected from dialogs – removing workers who exceeded a 10m threshold and discarding their data.

Dataset Splits. We follow the standard splits for the Matterport3D dataset [18] – dividing along environments. We construct four splits: train, val-seen, val-unseen, and test comprising 3,967/299/561/1,165 dialogs from 58/55/11/18 environments respectively. Val-seen contains new start locations for environments seen in train. Both val-unseen and test contain new environments. This allows us to assess generalization to new dialogs and to new environments separately in validation. Following best practices, the final locations of the

observer for the test set will not be released but we will provide an evaluation server where predicted localizations can be uploaded for scoring.

WAY includes dialogs in which the human Locator failed to accurately localize the Observer. In reviewing failed dialogs, we found human failures are often due to visual aliasing (e.g., across multiple floors), or are relatively close to the 3m threshold. We therefore expect that these dialogs still contain valid descriptions, especially when paired with the Observer's true location during training. In experiments when removing failed dialogs from the train set, accuracy did not significantly change.

5.3.2 WAY Dataset Analysis

Data Collection and Human Performance. In total, 174 unique workers participated in our tasks. On average each episode took 4 minutes and the average localization error is 3.17 meters. Overall, 72.5% of episodes where considered successful localizations at an error threshold of 3 meters. Each starting location has 3 annotations by separate randomly-paired Observer-Locator teams. In 40.9% of start locations, all 3 teams succeeded, in 36.3% 2, 18.5% 1, and 4.3% 0 teams succeeded. Figure 5.2 left shows a histogram of localization errors.

Why is it Difficult? Localization through dialog is a challenging task, even for humans. The teams success depends on the uniqueness of starting position, if and where the Observer chooses to navigate, and how discriminative the Locator's questions are. Additionally, people vary greatly in their ability to interpret maps, particularly when performing mental rotations and shifting perspective [112], which are both skills required to solve this task. We also observe that individual environments play a significant role in human error – as illustrated in Figure 5.2 right, larger buildings and buildings with multiple floors tend to have larger localization errors, as do buildings with multiple similar looking rooms (e.g., multiple bedrooms with similar decorations or office spaces with multiple conference rooms). The buildings with the highest and lowest error are shown in Figure 5.3.

a) Longest navigation distance b) Shortest navigation distance c) Highest localization error d) Lowest localization error

Figure 5.3: Environments with the largest/smallest mean navigation distance (a, b) and mean localization error (c, d). Observers tend to navigate more in featureless areas, such as the long corridor in (a). Localization error is highest in buildings with many repeated indistinguishable features, such as the cathedral with rows of pews in (c).

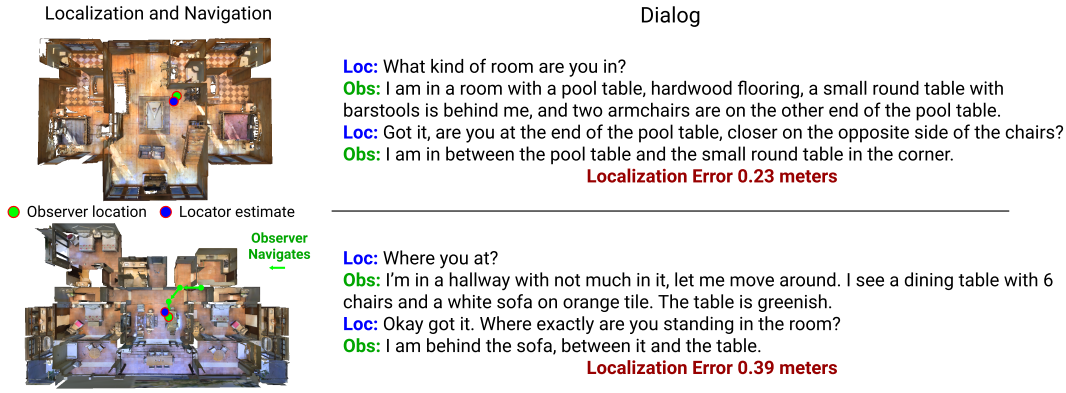

Figure 5.4: Examples from the dataset illustrating the Observer's location on the top-down map vs. the Locator's estimate (left) and the associated dialog (right). In the bottom example the Locator navigates to find a more discriminative location, which is a common feature of the dataset. The Observer navigates in 63% of episodes and the average navigation distance for these episodes is 3.4 steps (7.45 meters).

Characterizing WAY Dialogs. Figure 5.4 shows two example dialogs from WAY. These demonstrate a common trend – the Observer provides descriptions of their surroundings and then the Locator asks clarifying questions to refine the position. More difficult episodes require multiple rounds to narrow down the correct location and the Locator may ask the Observer to move or look for landmarks. On average, dialogs contain 5 messages and 61 words.

The Observer writes longer messages on average (19 words) compared to the Locator (9 words). This asymmetry follows from their respective roles. The Observer has first-person access to high-fidelity visual inputs and must describe their surroundings, *'In a kitchen with a long semicircular black counter-top along one wall. There is a black kind of rectangular*

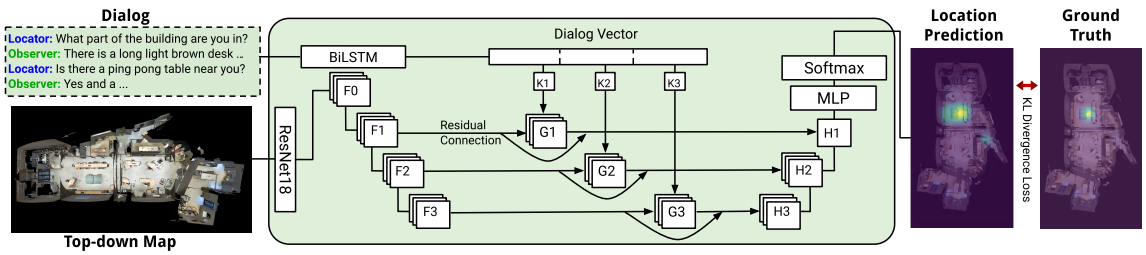

Figure 5.5: The 3-layer LingUNet-Skip architecture used to model the Localization from Embodied Dialog task.

table and greenish tiled floor.'. Meanwhile, the Locator sees a top-down view and uses messages to probe for discriminative details, *'Is it a round or rectangle table between the chairs?'*, or to prompt movement towards easier to discriminate spaces, *'Can you go to another main space?'*.

As the Locator has no information at the start of the episode, their first message is often a short prompt for the Observer to describe their surroundings, further lowering the average word count. Conversely, the Observer's reply is longer on average at 24 words. Both agent's have similar word counts for further messages as they refine the location. See the appendix for details on common utterances for both roles in the first two rounds of dialog.

Role of Navigation. Often the localization task can be made easier by having the Observer move to reduce uncertainty (see bottom example of Figure 5.4). This includes moving away from nondescript areas like hallways and moving to unambiguous locations. We observe at least one navigation step in 62.6% of episodes and an average of 2.12 steps. Episodes containing navigation have a significantly lower average localization error (2.70m) compared to those that did not (3.98m). We also observe the intuitive trend that larger environments elicit more navigation. The distributions for start and end locations for the most and least navigated environments in the appendix.

5.3.3 WHERE ARE YOU? Tasks

We now formalize the LED, EVD and CL tasks to provide a clear orientation for future work on the WAY dataset.

Localization from Embodied Dialog. The LED task is the following – given an episode comprised of a environment and human dialog – $(e, L_0, O_0, \ldots L_{T-1}, O_{T-1})$ – predict the Observer's final location p_T. This is a grounded natural language understanding task with pragmatic evaluations – localization error and accuracy at a variable threshold which we set to 3 meters. This task does not require navigation or text generation; instead, it mirrors AI-augmented localization applications. An example would be a system that listens to emergency services calls and provides a real time estimate of the caller's indoor location to aid the operator.

Embodied Visual Dialog. This task is to replace the Observer by an AI agent. Given a embodied first-person view of a 3D environment (see Observer view in Figure 5.1), and a partial history of dialog consisting of k Locator and $k-1$ Observer message pairs (L_0: *'describe your location.'*, O_0: *'I'm in a kitchen with black counters.'*, L_1 ...): predict the Observer agent's next navigational action and natural language message to the Locator. To evaluate the agent's navigation path, the error in the final location can be used along with path metrics such as nDTW [113]. Generated text can be evaluated against human responses using existing text similarity metrics.

Cooperative Localization. In this task, both the Observer and the Locator are modeled agents. Modeling the Locator agent requires goal-oriented dialog generation and confidence estimation to determine when to end the task by predicting the location of the Observer. Observer and Locator agents can be trained and evaluated independently using strategies similar to the EVD task, or evaluated as a team using localization accuracy as in LED.

5.4 Modeling Localization From Embodied Dialog

While the WAY dataset supports multiple tasks, we focus on Localization from Embodied Dialog as a first step. In LED, the goal is to predict the location of the Observer given a dialog exchange.

5.4.1 LED Model from Top-down Views

We model localization as a language-conditioned pixel-to-pixel prediction task – producing a probability distribution over positions in a top-down view of the environment. This choice mirrors the environment observations human Locators had during data collection, allowing straightforward comparison. However, future work need not be restricted to this choice and may leverage the panoramas or 3D reconstructions that Matterport3D provides.

Dialog Representation. Locator and Observer messages are tokenized using a standard toolkit [111]. The dialog is represented as a single sequence with identical 'start' and 'stop' tokens surrounding each message, and then encoded using a single-layer bidirectional LSTM with a 300 dimension hidden state. Word embeddings are initialized using GloVe [34] and finetuned end-to-end.

Environment Representation. The visual input to our model is the environment map which we scale to 780×455 pixels. We encode this map using a ResNet18 CNN [114] pretrained on ImageNet [115], discarding the 3 final conv layers and final fully-connected layer in order to output a 98×57 spatial map with feature dimension 128. Although the environment map is a top-down view which does not closely resemble ImageNet images, in initial experiments we found that using a pretrained and fixed CNN improved over training from scratch.

Language-Conditioned Pixel-to-Pixel Model. We adapt a language-conditioned pixel-to-pixel LingUNet [24] to fuse the dialog and environment representations. We refer to the adapted architecture as LingUNet-Skip. As illustrated in Figure 5.5, LingUNet is a con-

volutional encoder-decoder architecture. Additionally we introduce language-modulated skip-connections between corresponding convolution and deconvolution layers. Formally, the convolutional encoder produces feature maps $F_l = \text{Conv}(F_{l-1})$ beginning with the initial input F_0. Each feature map F_l is transformed by a 1×1 convolution with weights K_l predicted from the dialog encoding, i.e. $G_l = \text{Conv}_{K_l}(F_l)$. The language kernels K_l are linear transforms from components of the dialog representation split along the feature dimension. Finally, the deconvolution layers combine these transformed skip-connections and the output of the previous layer, such that $H_l = \text{Deconv}([H_{l+1}; (G_l + F_l)])$. There are three layers and the output of the final deconvolutional is processed by a MLP and a softmax to output a distribution over pixels.

Loss Function. We train the model to minimize the KL-divergence between the predicted location distribution and the ground-truth location, which we smooth by applying a Gaussian with standard deviation of 3m (matching the success criteria). During inference, the pixel with highest probability is selected as the final predicted location. For multi-story environments, each floor is processed independently during training. During inference only the ground truth final floor is processed. This is done to maintain accurate euclidean distance measurements for localization error as euclidean distance is not meaningful when measuring across points on floors in multi-story environments. This schema is used for all baselines experiments except for human locators who select from all floors.

5.4.2 Experimental Setup

Metrics. We evaluate performance using localization error (LE) defined as the Euclidean distance in meters between the predicted Observer location $\hat{p}_T$ and the Observer's actual terminal location p_T: LE $= ||p_T - \hat{p}_T||_2$. We also report a binary success metric that places a threshold k on the localization error – $\mathbb{1}(\text{LE} \leq k)$ – for 3m and 5m. The 3m threshold allows for about one viewpoint of error since viewpoints are on average 2.25m apart. We use euclidean distance for LE because the localization predictions of our LingUNet-Skip

model are not constrained to the navigation graph. Matterport building meshes contain holes and other errors around windows, mirrors and glass walls, which can be problematic when computing geodesic distances for points off the navigation graph.

Training and Implementation Details. Our LingUNet-Skip model is implemented in PyTorch [116]. Training the model involves optimizing around 16M parameters for 15–30 epochs, requiring $\sim$8 hours on a single GPU. We use the Adam optimizer [117] with a batch size of 10 and an initial learning rate of 0.001 and apply Dropout [118] in non-convolutional layers with $p = 0.5$. We tune hyperparameters based on val-unseen performance and report the checkpoint with the highest val-unseen Acc@3m. To reduce overfitting we apply color jitter, 180° rotation, and random cropping by 5% to the map during training.

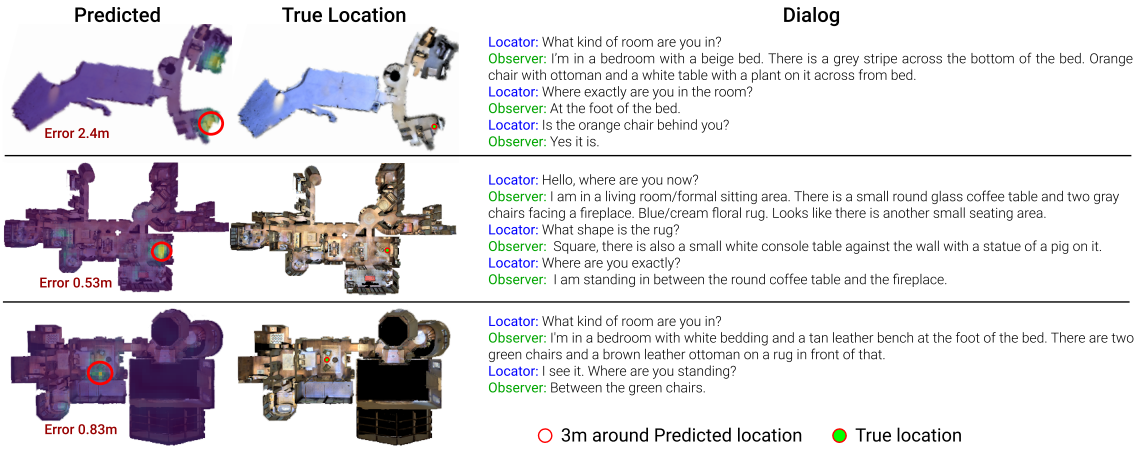

Figure 5.6: Examples of the predicted distribution versus the true location over top down maps of environment floors for dialogs in val-unseen. The red circle on the left represents the three meter threshold around the predicted localization. The green dot on the middle image represents the true location. The localization error in meters of the predicted location is shown in red.

Baselines. We consider a number of baselines and human performance to contextualize our results and analyze WAY:

– **Human Locator.** The average performance of AMT Locator workers as described in section 5.3.

– **Random.** Uniform random pixel selection.

– **Center.** Always selects the center coordinate.

– **Random Node.** Uniformly samples from Matterport3D node locations. This uses oracle knowledge about the test environments. While not a fair comparison, we include this to show the structural prior of the navigation graph which reduces the space of candidate locations.
– **Heuristic Driven.** For each dialog D_t in the validation splits we find the most similar dialog D_g in the training dataset based on BLEU score [119]. From the top-down map associated with D_g, a 3m x 3m patch is taken around the ground truth Observer location. We predict the location for D_t by convolving this patch with the top-down maps associated with D_t and selecting the most similar patch (according to Structural Similarity). The results (below) are only slightly better than random.

5.4.3 Results

Table 5.2 shows the performance of our LingUNet-Skip model and relevant baselines on the val-seen, val-unseen, and test splits of the WAY dataset.

Human and No-learning Baselines. Humans succeed 70.4% of the time in test environments. Notably, val-unseen environments are easier for humans (79.7%), see appendix for details. The Random Node baseline outperforms the pixel-wise Random setting (Acc@3m and Acc@5m for all splits) and this gap quantifies the bias in nav-graph positions. We find the Center baseline to be rather strong in terms of localization error, but not accuracy – wherein it lags behind our learned model significantly (Acc@3m and Acc@5m for all splits).

LingUNet-Skip outperforms baselines. Our LingUNet-Skip significantly outperforms the hand-crafted baselines in terms of accuracy at 3m – improving the best baseline, Center, by an absolute 10% (test) to 30% (val-seen and val-unseen) across splits (a 45-130% relative improvement). Despite this, it achieves higher localization error than the Center model for val-unseen and test. This is a consequence of our model occasionally being quite wrong despite its overall stronger localization performance. There remains a significant gap

Table 5.2: Comparison of our model with baselines and human performance on the LED task. We report average localization error (LE) and accuracy at 3 and 5 meters (all ± standard error). * denotes oracle access to Matterport3D node locations.

	val-seen			val-unseen			test		
Method	LE ↓	Acc@3m ↑	Acc@5m ↑	LE ↓	Acc@3m ↑	Acc@5m ↑	LE ↓	Acc@3m ↑	Acc@5m ↑
Human Locator	3.26±0.71	72.3±3.0	78.8±3.0	1.91±0.32	79.7±3.0	85.2±1.7	3.16±0.35	70 4±1.4	77.2±1.3
Random	12.39±0.31	5.4±0.9	15.0±1.3	10.18±0.16	7.0±0.7	21.3±1.1	13.10±0.17	6 6±0.5	15.2±0.7
Random Node*	8.27±0.44	18.1±2.2	37.8±2.7	10.44±0.31	15.8±1.1	29.0±1.4	13.19±0.32	12 8±0.7	24.9±0.9
Center	6.13±0.25	23.1±2.4	46.5±2.9	**4.90±0.12**	29.8±1.9	61.0±2.1	**6.71±0.14**	22 6±1.2	42.3±1.4
Heuristic	11.6±0.49	12.5±1.8	23.6±2.4	10.10±0.28	10.5±1.2	25.7±1.8	13.45±0.32	9 1±0.8	18.4±1.1
No Language	7.17±0.42	26.1±2.5	44.8±2.9	5.72±0.20	32.1±2.0	58.1±2.1	7.67±0.18	22 3±1.2	42.4±1.4
No Vision	11.36±0.46	9.4±1.7	18.4±2.2	8.58±0.20	7.8±1.1	22.1±1.8	11.62±0.23	7 7±0.8	18.3±1.1
LingUNet	**4.73±0.32**	**53.5±2.9**	**67.2±2.7**	5.01±0.19	**45.6±2.1**	**63.6±2.0**	7.32±0.22	**32.7±1.4**	**49.5±1.5**

between our model and human performance – especially on novel environments (70.4% vs 32.7% on test).

5.4.4 Ablations and Analysis

Table 5.3 reports detailed ablations of our LingUNet-Skip model. Following standard practice, we report performance on val-seen and val-unseen.

Navigation Nodes Prior We do not observe significant differences between val-seen (train environments) and val-unseen (new environments), which suggests the model is not memorizing the node locations. Even if the model did, learning this distribution would likely amount to free-space prediction which is a useful prior in localization.

Input Modality Ablations. No Vision explores the extent that linguistic priors can be exploited by LingUNet-Skip, while No Dialog does the same for visual priors. No Dialog beats the Center baseline (32.1% vs. 29.8% val-unseen Acc@3m) indicating that it has learned a visual centrality prior that is stronger than the center coordinate. This makes sense because some visual regions like nondescript hallways are less likely to contain terminal Observer locations. Both No Vision and No Dialog perform much worse than our full model (7.8% and 32.1% val-unseen Acc@3m vs. 45.6%), suggesting that the task is strongly multimodal.

Dialog Halves. First-half Dialog uses only the first half of dialog pairs, while Second-half

Table 5.3: Modality, modeling, and dialog ablations for our LingUNet-Skip model on the validation splits of WAY.

	val-seen		val-unseen	
	LE ↓	Acc@3m ↑	LE ↓	Acc@3m ↑
Full LingUNet-Skip Model	**4.73±0.32**	**53.5±2.9**	5.01±0.19	45 6±2.1
w/o Data Aug.	5.98±0.35	41.1±2.0	5.44±0.18	35 7±2.1
w/o Residual	5.26±0.33	47.5±2.9	4.74±0.17	43 1±2.1
No Dialog	7.17±0.42	26.1±2.5	5.72±0.20	32 1±2.0
First-half Dialog	5.06±0.33	50.5±2.8	**4.71±0.18**	**46.2±2.1**
Second-half Dialog	5.29±0.28	41.8±2.8	5.06±0.17	38 7±2.1
Observer-only	5.73±0.36	45.2±2.9	4.77±0.17	44 9±2.1
Locator-only	6.39±0.37	30.4±2.7	5.63±0.19	33 3±2.0
Shuffled Rounds	5.32±0.32	42.8±2.8	4.67±0.18	44 9±2.1

Dialog uses just the second half. Together, these examine whether the start or the end of a dialog is more salient to our model. We find that First-half Dialog performs marginally better than using the full dialog (46.2% vs 45.6% val-unseen Acc@3m) which we suspect is due to our model's failure to generalize second half dialog to unseen environments and problems handling long sequences. Further intuition for these results is that the first-half of the dialog contains coarser grained descriptions and discriminative statements ("I am in a kitchen"). The second-half of the dialog contains more fine grained descriptions (relative to individual referents in a room). Without the initial coarse localization, the second-half dialog is ungrounded and references to initial statements are not understood, therefore leading to poor performance.

Observer dialog is more influential. Observer-only ablates Locator dialog and Locator-only ablates Observer dialog. We find that Observer-only significantly outperforms Locator-only (44.9% vs. 33.3% val-unseen Acc@3m). This is an intuitive result as Locators in the WAY dataset commonly query the Observer for new information. We note that Observers were guided by the Locators in the collection process (e.g. *'What room are you in?'*), and

that ablating the Locator dialog does not remove this causal influence.

Shuffling Dialog Rounds. Shuffle Rounds considers the importance of the order of Locator-Observer dialog pairs by shuffling the rounds. Shuffling the rounds causes our LingUNet-Skip to drop just an absolute 0.7% val-unseen Acc@3m (2% relative).

Model Ablations. Finally, we ablate two model-related choices. Without data augmentation (w/o Data Aug.), our model drops 9.9% val-unseen Acc@3m (22% relative). Without the additional residual connection (w/o Residual), our model drops 2.5% val-unseen Acc@3m (5% relative).

5.5 Conclusion

In summary, we propose a new set of embodied localization tasks: Localization from Embodied Dialog - LED (localizing the Observer from dialog history), Embodied Visual Dialog - EVD (modeling the Observer), and Cooperative Localization - CL (modeling both agents). To support these tasks we introduce WHERE ARE YOU? a dataset containing $\sim$6k human dialogs from a cooperative localization scenario in a 3D environment. WAY is the first dataset to present extensive human dialog for an embodied localization task. On the LED task we present a LingUNet-Skip model which improves over simple baselines and model ablations but without taking full advantage of the second half of the dialog. We demonstrate that to model the LED task, no interaction is needed beyond the human annotations provided in the dataset.

CHAPTER 6

EMBODIED DIALOG GROUNDING FOR LOCALIZATION

6.1 Introduction

A key goal of AI is to develop embodied agents that can effectively communicate with humans and other agents using natural language. A basic capability for these agents is the ability to perceive and navigate through an environment and respond to instructions and questions about the space they are in. The recently-introduced Where Are You? (WAY) dataset [120] provides a setting for developing such a multi-modal and multi-agent paradigm. This dataset (collected via AMT) contains episodes of a localization scenario in which two agents communicate via turn-taking natural language dialog: An *Observer* agent moves through an unknown environment, while a *Locator* agent attempts to identify the Observer's location in a map.

The Observer produces descriptions such as *'I'm in a living room with a gray couch and blue armchairs. Behind me there is a door.'* and can respond to instructions and questions provided by the Locator: *'If you walk straight past the seating area, do you see a bathroom on your right?'* Via this dialog (and without access to the Observer's view of the scene), the Locator attempts to identify the Observer's location on a map (which is not available to the Observer). This is a complex task for which a successful localization requires accurate situational grounding and the production of relevant questions and instructions.

One of the benchmark tasks supported by WAY is *Localization via Embodied Dialog (LED)*. This task involves the development of a machine learning model that takes the dialog and a representation of the map as inputs, and outputs the final location of the Observer agent. LED is a first step towards developing a Locator agent. Two basic issues are to identify an effective map representation and deep learning architecture. The LED base-

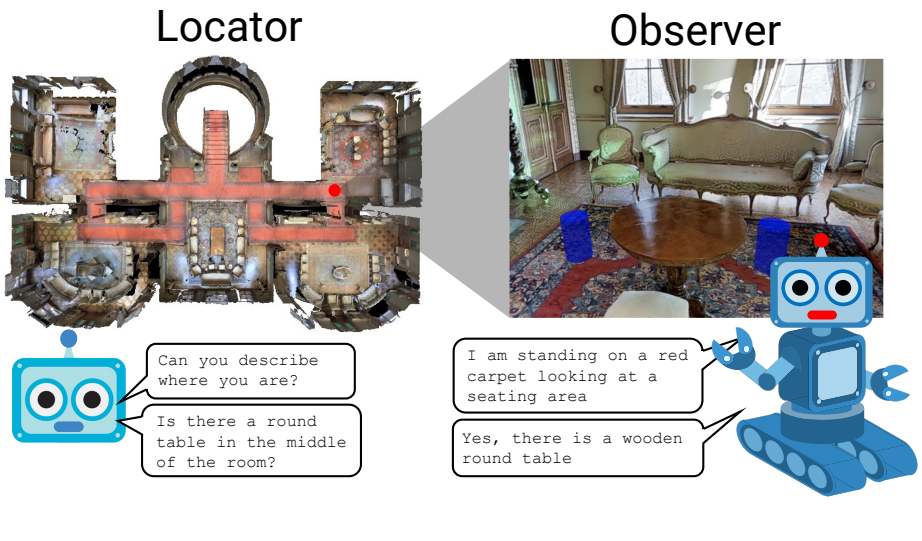

Figure 6.1: WAY Dataset Localization Scenario: The Locator has a map of the building and is trying to localize the Observer by asking questions and giving instructions. The Observer has a first person view and may navigate while responding to the Locator. The turn-taking dialog ends when the Locator predicts the Observer's position.

line from [120] uses 2D images of top down (birds-eye view) floor maps to represent the environment and an (x,y) location for the Observer.

This chapter explores multiple modeling approaches for the Localization via Embodied Dialog (LED) task. First, we discuss the possible map representations on which to produce a localization prediction and later demonstrate that a FPV panoramic navigation graph of the environments is more effective representation for modeling the task than the 2D top-down floor map proposed in Chapter 5. Chapter 5 contains previous baselines on the LED task, which approach the task as a prediction task over the pixels of 2D top down map. Specifically, in these baseline models, each environment is represented as a series of top down floor maps. Additionally the baselines use a simplistic approach for encoding the dialog using an LSTM without pre-training. These baselines are able to achieve some success, and are able to do so without online interaction within the environments, however they are far from solving the task or matching human performance. We propose that using

a panoramic graph representation of each environment is a more effective way to solve the LED task and a better proxy to the FPV memory that a human agent would have of an environment when solving a similar collaboration task. Additionally they are the same panoramic nodes the Navigator agent was viewing during the dialog creation, creating a direct mapping between the dialogs and the nodes.

Our work also investigates which multi-modal architecture is most effective for the LED task using graph-based maps. Additionally, previously LED baselines were trained only on the WAY dataset. This is a significant limitation given the small size of the WAY dataset in comparison to the abundance of visual grounding datasets that exist both in the image and Embodied AI domain. Drawing on previous success in other multi-modal tasks, we show that we can leverage these external datasets to pretrain an LED model to increase performance. Transfer learning from large-scale web data, where large models are pretrained on large datasets with simple training objectives and then fine-tuned on down stream tasks is an existing idea which has been shown to achieve high accuracy on many computer vision and natural language processing tasks. This has been particularly prevalent with transformer architectures which work best when trained with large amounts of data [83]. The WAY dataset contains only 6k dialog-localization episodes so we can supplement the data by using scraped web data and other multi-modal embodied datasets like Vision Language Navigation (VLN) [4] allowing our model to learn visual grounding concepts via other data and transfer this ability to the LED task. This type of transfer learning has already been proven to work for multi-modal language and vision tasks such as Visual Question Answering [85], Commonsense Reasoning [86], Natural Language Visual Reasoning [87], Image-Text Retrieval [88], and Visual-Dialog [7]. Additionally it has been shown to work in the Embodied AI domain via tasks like VLN [8]. We investigate how this pre-training schema can be extended to the task of localization and the linguistic complexity of dialog.

In this work we demonstrate this pretraining schema on a popular visio-lingustic model. Specifically, we adapt ViLBERT [121] for the LED task and show it outperforms all other

models. The ViLBERT model is a transformer model which contains a both a language and a vision encoder stream. The streams interact via co-attention allowing for inputs of each modality to be conditioned on the other. The output of the model, as we use it, produces an alignment score between the vision and language inputs. Additionally we propose a new set of strong baselines to compare against which also model the localization over the graph nodes.

Contributions: To summarize:

1. We demonstrate an LED approach using navigation graphs to represent the environment.
2. We present LED-Bert, a visiolinguistic transformer model which scores alignment between graph nodes and dialogs, and we develop an effective pretraining strategy.
3. We show that LED-Bert outperforms all baselines, increasing accuracy at 0m by 8.21 absolute percent on the test split.

6.2 Approach

We propose to use multi-modal architectures which can use cross-modal attention between the dialog and visual map to perform accurate grounded localization's.

6.2.1 Environment Representation for Localization

A key challenge in the LED task is that environments often have multiple rooms with extremely similar attributes, i.e. multiple bedrooms with the same furniture. Therefore a successful model must be able to visually ground fine-grained attributes. Strong generalizability is also required in order to generalize to unseen test environments. The LED baseline in Chapter 5 approaches localization as a language-conditioned pixel-to-pixel prediction task – producing a probability distribution over positions in a top-down view of the environment Figure 6.2. This choice is justified by the fact that it mirrors the observations that human Locators had access to during data collection, allowing for a straightforward

comparison. However, this does not address the question of what representation is optimal for localization.

The real-world proxy presented as motivation for the embodied localization task is to imagine a lost person in a building communicating with another person who is very knowledgeable about the building and is trying to find them. While there are multiple theories in psychology of how humans cognitively represent space, it is almost certain that the knowledgeable friend experienced the environment from a first person view (FPV). Therefore using a FPV map representation is a more logical choice then a 2D top down map representation for modeling the LED problem. The FPV map representation we propose to use is the panoramic-RGB graphs of the Matterport environments [18] as seen in Part B, Figure 6.2. The Observer agent traverses these same navigation graphs during data collection, which may result in a strong alignment between the dialog and the nodes. Using this approach, the LED task can be framed as a prediction problem over the possible nodes in the navigation graph. At inference time, this can be accomplished by producing an alignment score between each node in the test environment and the test dialog, and then returning the node with the highest score as the predicted observer location.

6.2.2 ViLBERT

ViLBERT [121] is a multi-modal transformer that extends the BERT architecture to learn joint visio-linguistic representations. There are other similar approaches to ViLBERT [122, 123, 124, 125, 126]. Training this network requires paired image-text data, and ViLBERT specifically uses the Conceptual Captions dataset [127] which is a large dataset of images from the web paired with an alt-text. ViLBERT is constructed of two transformer encoding streams, one for visual inputs and one for text inputs. Both of these streams are constructed using a standard BERT-BASE [83] backbone. The input for the text stream is then text tokens, identical to BERT. The visual tokens for the visual stream is a sequence of image regions which is generated by an object detector pretrained on Visual Genome. We can

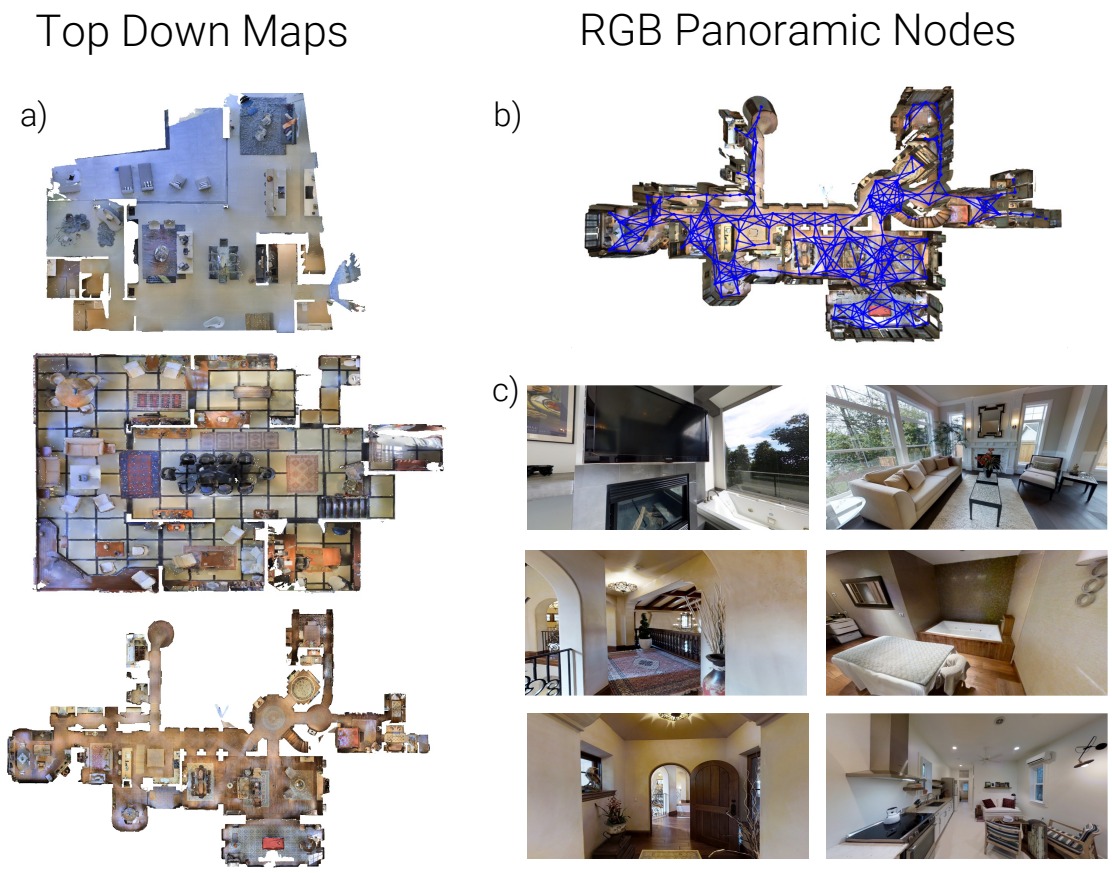

Figure 6.2: Examples of the types of map representations of the Matterport3D [18] indoor environments which can be used for the Localization via Embodied Dialogue task. Part A shows the top down floor maps used in the original LED paper. Part B shows an overlay of the navigation graph of panoramic nodes over the top down map, note the lines represent traversability between nodes and the circles represent the panoramic node location. Part C shows examples of the FPV panoramic nodes in different environments. Note each of these images are mapped to a node in a connectivity graph for the respective environment.

then think of the input to ViLBERT being a sequence of visual and textual tokens which are not concatenated and only enter their respective streams.

The two streams then interact using co-attention layers which are implemented by swapping the key and value matrices between the visual and textual encoder streams for certain layers, this is illustrated in Figure 6.3. The idea of a co-attention layer is to attend to one modality via a conditioning on the other modality. Allowing for attention over image regions given the corresponding text input and vise versa. Masked multi-modal modelling and multi-modal alignment tasks are used to train ViLBERT and are used in the pretraining

and fine-tuning of LED-BERT. Masked multi-modal modelling works in a similar fashion to masked language modeling in BERT. The multi-modal alignment objective trains ViLBERT to determine if a input image-text pair are well aligned and matched. This alignment objective can be directly extended to matching dialogs and node pairs.

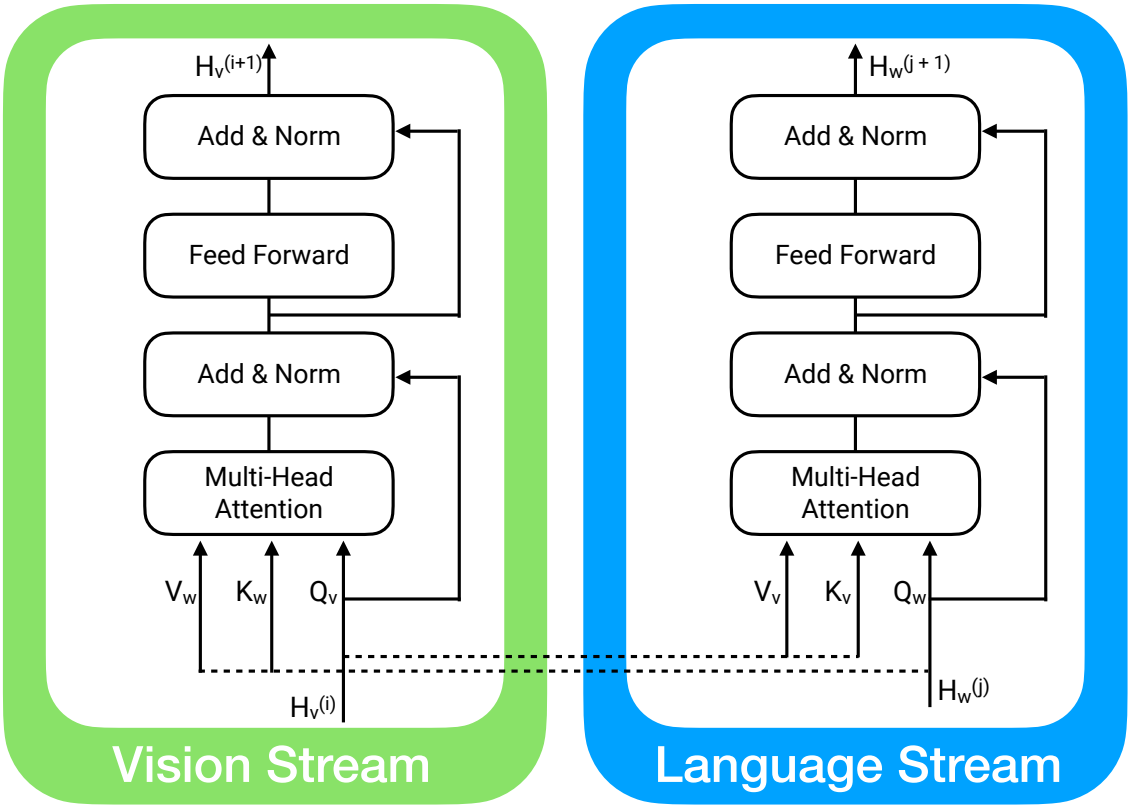

Figure 6.3: Illustration of the co-attention layer introduced in ViLBERT [121] and used in LED-Bert.

6.2.3 Adapting ViLBERT for LED

To formalize the graph based LED task, we consider a function f that maps a node location n and a dialog x to a compatibility score $f(n, x)$. We model $f(n, x)$ using a visiolinguistic transformer-based model we denote as LED-Bert, shown in the Appendix, Figure Figure 6.4. The architecture of LED-Bert is structurally similar to ViLBERT and VLN-Bert [8], but with some key differences due to our need to ground dialog and fine-tune on the relatively small WAY dataset. We followed this structure in order to be enable

transferring the visual grounding learned via these models from disembodied large-scale web data and similar embodied grounding tasks. This allows us to initialize the majority of the LED-Bert using pretrained weights. We represent each panoramic node I as a set of image regions $r_1, ..., r_k$. Let an dialog x be a sequence of tokens $w1, ..., w_L$. Then for a given dialog-node pair for LED-Bert as the input sequence:

$$\langle \texttt{IMG} \rangle r_1, ..., r_K \langle \texttt{CLS} \rangle w_1, ..., w_L \langle \texttt{SEP} \rangle \tag{6.1}$$

where IMG, CLS, and SEP are special tokens.

To train LED-BERT for path selection, we consider a 5-way multiple-choice task. Given an dialog x, we randomly sample four nodes from the current environment and sample the node which is the true location of the Observer agent $N = (n_1, n_2, n_3, n_4, n_5)$. LED-BERT is run for each dialog-node pair and the final representations for the CLS and IMG token are extracted and denoted as hCLS and hIMG. To compute a compatibility score s_i between these two representations we use the following equation:

$$si = f(n_i, x) = W(h^{(i)}CLS \odot h^{(i)}IMG) \tag{6.2}$$

where $\odot$ denotes element-wise multiplication and W is a learned transformation matrix. The scores are normalized via a softmax and are supervised with cross-entropy loss. During inference time each node in the environment is run through LED-BERT and the nodes are sorted by there compatibility score s_i and the node with the highest score is predicted as the location of the Observer agent.

Transformer models are by nature invariant to sequence order and they only model interactions between inputs as a function of their values [128]. Therefore it is standard to re-introduce order information by adding positional embeddings for each input token. For the dialog tokens this is straight forward and is simple an index sequence order encoding. However the panoramic node visual tokens have a more complicated positional encoding,

as the panorama is broken up into image regions. The visual positional information is very important for encoding spatial relationships between objects and for scene understanding as a whole. For instance consider the a question the Locator might as *'Are you located to the right of the blue couch?'* This question will require information about which part of the panorama the couch is located in. To address this, we follow the VLN-BERT [8] strategy of encoding the spatial location of each image region, in terms of its location in the panorama (top-left and bottom-right corners in normalized coordinates as well as area of the image covered), its elevation relative to the horizon and all angles are encoded as [cos(θ),sin(θ)]. The resulting 11-dimensional vector S is projected into 2048 dimensions using a learned projection W^S.

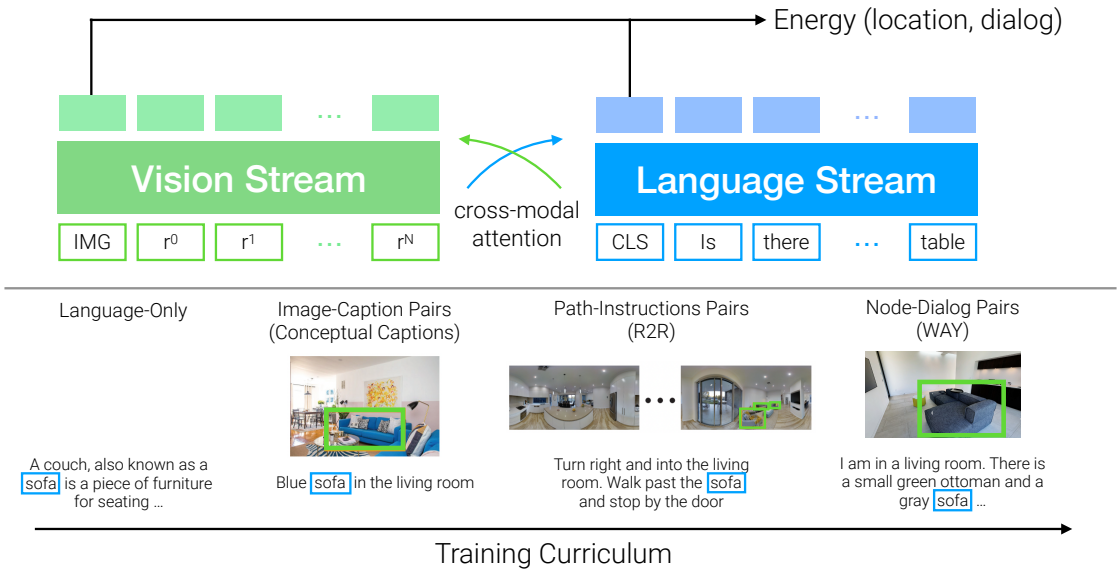

Figure 6.4: We propose the LED-BERT for the LED task. The language stream of the model is first pretrained on English Wikipedia and the BooksCorpus [84] datasets. Second, both streams of the model are trained on the Conceptual Captions [127] dataset. Third, both streams are train on the path-instruction pairs of the Room2Room dataset [4]. Finally we fine-tune the model over the node-dialog pairs of the WAY dataset [120].

6.2.4 Training Procedure for LED-BERT

LED-BERT can be trained from scratch using the WAY dataset however due to the small size (6k episodes) of the WAY dataset and since large-transformer models work best on

large amounts of data we follow a pretraining set up. Therefore we follow the inspiration of prior work [8, 7, 121] which does extensive pretraining for multi-modal transformers using large scale web-data. The pipeline for pretraining has 4 stages and is also visualized in Figure 6.4.

Stage 1 - Language Only: The language stream of LED-BERT is initialized with the BERT-BASE weights which is trained on Wikipedia and BookCorpus [84] using the masked language modeling and next sentence prediction training objectives. This allows the model to be initialized with a large degree of natural language understanding.

Stage 2 - Dual Streams and Visual Grounding: Following [121], both streams are trained using the Conceptual Captions dataset of image-caption pairs. During this stage the model is trained with the masked language modeling, masked image modeling and multi-modal alignment objectives. This stage of training initializes the cross modal attention layers and teaches the model language grounding over visual features.

Stage 3 - Dual Streams and Visual Grounding: Following [8] we use the navigation path - instruction pairs from the R2R dataset [4] to train the model using masked language modeling and masked vision modeling. This allows the model to map objects and visual concepts from the path directly to the text instructions. For example the text instructions may contain *'Stop at the [MASK].'* and the model must predict the word couch based on the visual path.

Stage 4 - Fine Tuning for Node Localization: To train LED-BERT for localization, we consider the task as a selection task over the possible nodes in the graph, on average there are 117.32 nodes, with the largest environment containing 345 nodes. We run LED-BERT on each node-dialog pair and extract the final representations for each stream, denoted as $h_{\texttt{CLS}}$ and $h_{\texttt{IMG}}$, using these we compute a compatibility score s by doing element-wise multiplication of the two vectors and passing them through a single linear layer. The scores are normalized via a softmax layer and then supervised using a cross-entropy loss against a one-hot vector with a mass at the ground truth node. There is no need to do hard negative

examples as each node in the entire environment is considered.

6.3 Experiments

6.3.1 Baselines

We propose multiple strong baseline methods to compare the LED-BERT architecture against. All approaches will use the panoramic based maps to ensure the same the prediction space of all panoramic nodes in an environment.

Human Performance: Using the average performance of AMT Locator workers from the WAY dataset. We snap the human prediction over the top down map to the nearest node to where the AMT worker selected.

Random: Selecting a random node from the environment as the predicted location for each test episode.

Joint Embedding: This baseline learns a common embedding space between the dialogs and corresponding node locations. This simple design allows us to easily create a naive baseline for the LED task setup using the panoramic nodes. In this baseline we propose to use visual features from a ResNet152 [114] pretrained on Places 365 [102]. Each panoramic node is represented by 36 image patches, image features are extracted for each patch and represent the visual features of the node. We experiment with three types of joint embedding architectures. The dialog for these models are all encoded similar to the Ling-U-Net skip model described in Chapter 5. The Locator and Observer messages are tokenized using a standard toolkit [111]. The dialog is represented as a single sequence with identical 'start' and 'stop' tokens surrounding each message, and then encoded using a single-layer bidirectional LSTM. Word embeddings are initialized using GloVe [129] and fine tuned end-to-end. In the first model called the 'late-fusion model', the LSTM has a 2048 dimension hidden state and the node features are down-sampled using self attention to be of size 2048, the visual and dialog features are fused through late fusion passed through a two-layer MLP and softmax and the output is a prediction over the possible nodes in

the environment. In the 'attention model', the visual and dialog features are fused instead through top-down bottom up attention, the final layers of the model are also an MLP and softmax. In the 'attention over history model', there is a separate LSTM to encode dialog history and a another LSTM to encode the current message. The using dialog history features attention is applied over the visual features to reduced their size and then the current message features and visual features are fused through late fusion, followed by an MLP and softmax.

Graph Convolutional Network All other baselines and LED-BERT discard edge information and only make a single prediction over the entire dialog. Therefore we propose to experiment with Graph Convolutional Network (GCN) [130] to model the LED task and incorporate edge information. Note the graph inputted to the GCN is the navigation graph that the observer navigated through as the dialog was created. Nodes in the graph are connected via edges that denote navigational connectivity. Edge information therefore consists of which nodes are connected and the proximity of the connected nodes via pose transformation. In many of the localization episodes the Observer agent navigates a few meters which leads to the dialog referring to multiple locations that are spatially close together. For example, a dialog could be referring to a single bedroom in a house and discussing multiple places in the bedroom that each correspond to a different node. We posit that being able to aggregate visual information of the connected nodes will allow for more informed and accurate localization predictions. GCNs provide a way to do this via using edge information. GNNs have been used successfully in the embodied settings of visual navigation [65, 131] as well as the multi-modal settings of video and image captioning [77, 78]. In Chapter 5 of this thesis, we showed the NRNS approach for image-navigation, which used an augmented Graph Attention Network [100] to estimate the distance to goal of different ndoes. Distance estimation was modeled by doing weighted aggregation of neighboring node information. We use a similar architecture and a similar graph representation to NRNS with nodes containing attributes of visual features and edge attributes containing the pose transformation

Table 6.1: Comparison of the LED-BERT model with baselines and human performance on the LED task. We report average localization error (LE) and accuracy at k meters (all $\pm$ standard error).

	val-seen			val-unseen			test		
Method	LE ↓	Acc@0m ↑	Acc@5m ↑	LE ↓	Acc@0m ↑	Acc@5m ↑	LE ↓	Acc@0m ↑	Acc@5m ↑
Human Locator	6.00	47.87	77.38	3.20	56.13	83.42	5.89	44 92	75.00
Random Node	20.8	0.33	10.82	18.61	1.9	11.05	20.93	0 92	11.00
Late Fusion	12.56	17.38	47.54	12.87	7.77	34.37	15.86	8 92	32.75
Attention Model	9.83	18.36	56.07	10.93	10.54	41.11	14.96	6 92	34.42
Attention over History Model	11.64	21.64	49.18	11.44	10.02	43.18	14.98	7 14	33.68
Graph Convolutional Network	10.95	19.67	59.13	9.10	8.64	46.99	14.32	9 46	35.1
LED-BERT	**9.04**	**25.57**	**60.66**	**8.82**	**21.07**	**52.5**	**11.12**	**17.67**	**51.67**

between connected nodes. A key difference in the graph structure between this architecture and NRNS is that in the proposed work the topological graphs are pre-defined [97] and there are no unexplored nodes. The goal of the GCN architecture is to model the relational information between the nodes of the graph and the localization dialog in order to produce a probability distribution of localization likelihood over the nodes in the graph.

6.3.2 Metrics

We propose to evaluate the localization error (LE) of our models using geodesic distance instead of euclidean distance as used in [120]. Geodesic distance will be more meaningful than euclidean distance for determining error across rooms and across floors in multi-story environments. In addition to localization error we also report a binary success metric that places a threshold k on the localization error. Accuracy (Acc) at 0 meters indicates the correct node was predicted. Accuracy at k meters indicates that the node predicted was within k meters of the true location. This is helpful to understand the precision of the locator model.

6.3.3 Results

Table Table 6.1 shows the performance of our LED-BERT model and relevant baselines on the val-seen, val-unseen, and test splits of the WAY dataset.

Human and No-learning Baselines. Humans succeed 44.92% of the time in test environments at 0 meters, this shows it is a difficult task.

Attention and History increase performance. For the similar architectures, using bottom-up and top-down attention increases performance, additionally separating the encoders for the current message from the dialog history increases performance as well.

Graph Networks see slight improvement. Graph networks increase performance further. This could be attributed to the utilization of edge information. Graph networks however do not have a straight forward pretraining schema for this task.

LED-BERT outperforms all baselines. LED-BERT significantly outperforms the other cross-modal modeling baselines in terms of both accuracy and localization error – improving the best baseline, Graph Convolutional Network (GCN), by an absolute 7.54% (test) to 12.43% (val-seen and val-unseen). There remains a gap between our model and human performance – especially on novel environments (-% vs -% on test).

LED-BERT outperforms graph networks. Despite the intuition modeling with edge information would allow aggregation of visual features to allow for better localization predictions under episodes with navigation, LED-BERT which uses no edge information outperformed the GCN. We believe this can be partially attributed to the large scale pretraining for the LED-BERT network. Using an adapted ViLBert model allowed for direct transfer learning from multiple data sources including large language corpora, image-caption pairs, and vision-language navigation episodes. The GCN model however does not have a straight forward way to do pre-training and therefore the dialog encoder and graph encoder are trained from scratch which then suffers the limitation of the small size of the WAY dataset and is susceptible to over-fitting to the training environments.

6.4 Conclusion

In summary, we propose a viso-linguistic transformer, LED-BERT, for the Localization from Embodied Dialog task and instantiate a new version of the LED task which does

localization over the navigation graph. We demonstrate a pre-training schema for LED-BERT which utilizes large scale web-data as well as other multi-modal embodied AI task data to learn the visual grounding required for successful localization's in LED. We show LED-BERT is able to achieve SOTA performance and outperform other learned baselines by a significant margin.

CHAPTER 7

SPATIAL INSTRUCTIONS FOR NAVIGATION AND LOCALIZATION

7.1 Introduction

In the previous chapter we observed that LED-BERT was able to outperform a Graph Convolutional Network (GCN) on the Localization Via Embodied Dialog task despite being given no edge information or information about neighboring nodes. We attribute this to the pre-training of LED-BERT using large scale multi-modal web data. LED-BERT works by taking in a single panorama and dialog pair and predicting an alignment score between them. The panorama in the environment with the highest alignment to the dialog is predicted as the location of the agent. The LED-BERT architecture is adapted from the multi-modal ViLBERT transformer architecture. To introduce order to the segmented panorama when inputted to the model, each segment additionally receives a positional embedding. In many instances the dialog in the WAY dataset episodes refers to spatial layout of objects in the environment and the agents position to these objects. For example " do you see a green sofa in front of you and a exit on your right?" Due to the nature of the nodes being panoramic, learning the meaning of "in front" as well as then relating this to the "right" direction is challenging. In this chapter we sought to understand to what degree the model is learning spatial and directional words and how these words impact the performance of the model. LED-BERT is based on the VLN-BERT model which also takes in panoramic nodes, so we chose to run these experiments in parallel with the Vision Language Navigation (VLN) task. VLN provides an interesting test bed because navigation instructions contain more spatial and directional words than the WAY dialogs as well as VLN-BERT takes in a sequence of panoramas and must learn spatial information in and between panoramas in this sequence. The analysis in this chapter provides insight to using

visio-linguistic transformer models for embodied AI tasks.

Vision Language Navigation (VLN) is the task of having a robot navigate via following human instructions such as *'Leave the bathroom and walk to the right...'*. The common benchmarks for VLN [4, 132, 11, 133, 5] are conducted in a simulated environments such as Matterport3D (MP3D) [18] with human annotated instructions. The task is complex in similar ways to the LED task, it requires accurate visual grounding of objects and visual descriptions provided in the instructions and it requires understanding of spatial information to ground instructions such as *'walk to the right'* into actions. The common benchmark called R2R [4] is set over the same discrete MP3D environment as the LED task [3], which is constructed of panoramic nodes which have navigational connectivity to neighboring nodes. The action space therefore for the navigating agent is the possible neighboring nodes or to stop the navigation. This set up has led to two distinct approaches to the VLN task, one which is discriminative [8, 134] and one which is generative [4, 135, 136, 137]. In the discriminative approach, using beam search from the given starting location, up to 30 possible paths are generated and a path is selected using a discriminative path selection model. In the generative approach the agent is placed at the starting location and recursively selects the next node to navigate to, until selecting the stop action.

The modeling for both the generative and discriminative approaches have seen large success by using multi-modal transformer models which leverage large-scale pre-training [8, 134, 136]. Pre-training data usually consists of large scale web data [127, 7, 121, 138] containing image-text pairs to learn visual grounding as well as large text corpora [84] to learn linguistic semantics. Success of models for both approaches are primarily measured in terms of Success Rate (SR) which measures the percentage of selected paths that stop within 3m of the goal. Additionally, models are evaluated using success rate weighted by path length (SPL), which provides a measure of success rate normalized by the ratio between the length of the shortest path and the selected path.

Other works [139, 140, 141, 142, 143] have tried to investigate what exactly the gen-

erative models are attending too and where the failure modes are. These works find that the generative VLN agents refer to object tokens and direction tokens in the instruction to make predictions. However there has been a lack of diagnostic evaluation over the discriminative models. This evaluation is important to see if the models are too reliant on one type of token in instructions and for understanding of how to improve performance. In this chapter we outline a simple method via token masking to understand how different types of part of speech and object vs direction tokens are used by discriminative models. Through our experiments we find that the discriminative models rely most heavily on nouns almost disregarding direction tokens and other parts of speech. Additionally, we find that changing direction tokens while holding nouns tokens constant leads to no effect on the model predictions. This highlights a large limitations of these models as they are not capturing large amounts of the available information for predictions. We additionally find this same phenomena happens in the transformer based Localization Via Embodied dialog models which treats the localization problem as a discriminative prediction problem over the panoramic nodes of the navigation graph.

Contributions:

1. We develop a series of experiments to diagnose which types of tokens influence VLN models predictions and to what degree. Using these experiments we find that generative VLN models equally attend to object and spatial tokens, however discriminative VLN models only attend to object tokens completely disregarding spatial tokens and all parts of speech other than nouns.
2. We experiments with adding a training stage which uses additionally masked language loss over only spatial tokens and a shuffle visual tokens to create additional hard negatives and find this increases the dependency of the model on spatial words as well as increase overall performance of the model.
3. We experiment with adding programmatically generated instruction-path pairs to the R2R dataset. These generated instructions contain few object words to the training

Table 7.1: Comparison of the language between the common VLN benchmark datasets and the related WAY dataset. Compares the size of the datasets and density of different POS.

	Reverie [133]	RXR [132]	R2R [4]	WAY [3]
Dataset Size	21702	25368	21582	7029
Vocab Size	4815	3779	3999	5888
Avg Num Tokens	18.3073	97.2956	29.3665	78.6463
Noun Density	0.3155	0.2104	0.2775	0.2204
Adj Density	0.0505	0.0615	0.0461	0.0703
Verb Density	0.1163	0.1690	0.1222	0.1299
Left-Right Density	0.0487	0.0492	0.0664	0.0299

set and training on them increases dependency on the spatial words.

4. We hope that these findings which reveal the limitations of current VLN models will lead to new research.

7.2 Comparison of Dataset Language

In order to understand the tokens that an navigational agent can attend to in embodied navigation tasks, Table 7.1 compares the language of instructions against existing VLN datasets and the Where Are You (WAY) dataset. Specifically we compare: dataset size, vocab size, average text length per episode, and density of different parts of speech (POS) and spatial tokens per episode. Vocab size was determined by the total number of unique words. We used the [111] POS tagger to calculate the POS densities over the text in each dataset. We note that the RxR task has significantly longer instructions than any of the other datasets. When looking at the density of different parts of speech, we find that all datasets have a high density of nouns and lower density of adjectives and verbs. There is also a low density of spatial words like “left” and “right”, which is to be expected as they are maybe only used a few times in an instruction however we see the navigational datasets have a much higher density than the localization dataset (WAY).

7.3 Masking Experiments

Modeling for both the generative and discriminative approaches have seen large success by using multi-modal transformer models which leverage large-scale pre-training [8, 134, 136] and in this work we seek to investigate what exactly what the top performing models are attending too for both the generative and discriminative approaches. In order to do this we devise an experiment set up in which we train the models with their normal training procedure and then during evaluation mask out different parts of speech and test performance. This allows us to get an understanding of how much the VLN model is relying on different types of tokens to make the navigational predictions. We examine 5 different masking criterion: nouns, verbs, adjectives, left-right, spatial words. We add an additional experiment in which we swap all 'left' and 'right' tokens and report performance. There is no standard list of spatial words so via qualitative analysis of the instructions amongst the standard VLN benchmarks we define the following words as spatial words: *'right, left, straight, near, front, through, down, up, between, past, stop, surround'*. The intuition behind these experiments is that if the model equally attends to all types of tokens performance should drop equally between different tokens being masked. Additionally we assume that instructional phrases such as 'take a left' being changed to 'take a right', should have a significant impact on performance of a navigational agent. We specifically focus our experiments on the R2R dataset for the VLN task as it is the most widely used. Figure 7.2 shows examples of the VLN instructions of the R2R dataset and how the navigation instructions are tokenized, part of speech tagged and masked for different criterion.

We wish to examine top performing models so we pick the top discriminative models: VLN-BERT [8] and AirBert [134] and the top performing generative model Recurrent-VLN-BERT [136]. These models are all multi-modal transformers which use large-scale pre-training via web data. AirBert is an extension of the VLN-BERT model and leverages and additional loss and more pre-training data scrapped from AirB&B to increase perfor-

Table 7.2: Results of the token type masking experiments across generative and discriminative VLN methods and a discriminative LED method. The results are shown in terms of Success Rate (SR). SR of VLN models measures the percentage of selected paths that stop within 3m of the goal. SR of LED models measures the percentage of locations selected that are within 0m of the true agent location. The first column is the models performance with no augmentation to the input language.

	masking - criteria						
Model	No Masking	Swap	Left-Right	Spatial Words	Adjectives	Nouns	Verbs
LED-Bert	0.2245	0.2228	0.2228	0.2159	0.2090	0.0708	0.2159
VLN-Bert [8]	0.5926	0.5960	0.5951	0.5866	0.5922	0.4491	0.5939
AirBert [134]	0.6645	0.6603	0.6582	0.6458	0.6582	0.4994	0.6594
Recurrent-VLN-Bert [136]	0.6275	0.4670	0.5466	0.4964	0.5917	0.4393	0.5802

mance. Additionally we include experiments over LED-BERT as it is also an extension of VLN-BERT but is trained and tested on a localization dataset rather than a navigation dataset. This allows us intuition on if our findings can be attributed to the model design or the individual datasets. Figure Figure 7.1 illustrates the comparison between the LED and VLN BERT based models as well as their training procedure.

Table 7.2 shows the results of the masking experiments for the VLN models on the val-unseen split of the R2R dataset, as well as the results of the LED-BERT model on the val-unseen split of the WAY dataset. All result shown are in terms of Success Rate (SR). For the VLN task SR measures the percentage of selected paths that stop within 3m of the goal. For the LED task SR measures the percentage of locations selected that are within 0m of the true agent location. The SR between the LED and VLN tasks are not comparable however the pattern across the drop in performance between masking different token types is comparable. Note that LED-Bert, VLN-Bert, and Airbert are discriminative models in that they are predicting alignment between either a location (LED) or given navigation paths (VLN) and an a text instruction (VLN) or dialog (LED). Recurrent-VLN-Bert is generative in that it predicts each node in the navigation path in a iterative fashion until predicting to stop navigating.

Discriminative models rely significantly on nouns. We observe in Table 7.2 that the dis-

criminative models only suffer in performance when noun tokens are masked. Performance less than 2% for all other types of token masking. In fact we even see performance for VLN-Bert slightly increase by up to .0034% when the 'right' and 'left' tokens are swapped to their antonym or when they are masked out. These results indicate the models are heavily relying on noun tokens while disregarding other information. The lack of dependency on spatial words is especially concerning as they are an integral part to what constitutes a directional instruction for navigation. As the discriminative VLN models are presented with the entire navigation path when predicting alignment, we hypothesize that the models are relying on noun words to do pattern matching and disregarding other visual features and positional panoramic information of the path.

R2R paths have a high number of counterfactuals. One reason that little effect on performance would be seen when swapping 'left' and 'right' tokens or masking them out, could be that many paths simply don't have a counterfactual. In other words, when an instruction states to 'turn right' it is possible there is no option to turn left in the discrete panoramic node setting of MP3D [18] of the R2R dataset. For example, imagine the navigation agent exits a room to find a hallway where only neighboring nodes are to the right of the agent. To discredit this hypothesis we first look at each turn in the R2R paths for the val-unseen data split. We determine any turn to be when the heading of the agent changes by over 30°degrees between two nodes in the path. We find there are an average of 1.67 turns per episode. Then for each node where the agent makes a turn we determine if there are any neighboring nodes which the agent could have navigated to instead, which turn in the opposite way or go straight. We call these nodes counterfactuals and we find that per turn there is an average of 1.62 counterfactuals. This discredits the possibility that directional tokens do not serve a significant role in the VLN task.

Generative VLN models rely on multiple types of tokens. We observe in the masking experiments that there are significant drops in performance of the Recurrent-VLN-Bert model when masking out any type of token. Nouns tokens still have the highest effect

on performance, however, this most this cannot be disentangled from the fact that noun tokens have the highest density in the input instructions of the types of the tested tokens, see Table 7.1. We additionally find that swapping the direction tokens leads to a sharp drop in performance which is almost at that of masking nouns.

7.4 Training via Passive Data

Discriminative transformer based models for VLN rely heavily on noun tokens while disregarding other types of tokens. Directional and spatial tokens provide significant information that these models are currently not taking advantage of. To try and increase the performance of the model and shift dependency on noun tokens we try multiple strategies including creation of hard negatives, an additional training step and training with automatically generated passive data. In Chapter 4 we saw how navigational agents could be trained via passive data for the task of image-goal navigation. In that setup the passive data consisted of a video of a fly-through paired with positional information for each frame of the video. To combat model reliance on tokens that describe object and rooms, we seek to inject new data during the training stage on R2R which contains minimal references to object and room names. We generate the new paths and the instructions programmatically allowing us to forgo the need for additional human annotations.

To generate the additional navigational paths we use a similar strategy to that of [51]. We first sampled start and goal location pairs in the MP3D training environments and then found the shortest path on the scene's navigation graph. We discarded any paths that were contained in the R2R dataset and any paths with less than 5 edges and over 10 edges. We then generated instructions over the paths. If a path had a turn of over 120°degrees we generated the instruction "turn around" otherwise any turn over 30°degrees generated the instruction "turn (left—right)". Otherwise instructions such as "go straight", "go forward", "continue straight", etc for "x meters" were generated, where x was determined to meters for the straight region of the path. If the path navigated over stairs we generated an in-

struction to “go (up—down) the stairs”. Each generated instruction ended with a “stop” or “wait here” command. In total we generate an additional 6k instruction-path pairs over the training environments of MP3D. For example a generated instruction in our dataset is *‘Go forward and walk 3 meters. Turn right, and walk one meter. Stop.’*

We then add the new generated path-instruction pairs to the training split of the R2R dataset. We retrain the VLN-BERT model with the additional data for stage 3 and then fine-tune only on the original RxR train split. In stage 3 of training the model is being trained by the masked multi-modal modeling objectives. The results of the model trained with additional passive data is shown in Table 7.3.

7.5 Data Augmentation

The VLN-BERT model is trained using masked multimodal modeling (MLM) and multi-modal alignment training objectives. For MLM a randomly selected set of the input text and image tokens are masked and the model must predict the original tokens given the surrounding context. As demonstrated by ViLEBERT demonstrated that for image regions, this can be done by predicting a distribution over object classes present in the masked region. Masked text tokens however are handled the same as in BERT where the original token is predicted directly. The MLM objective is used during stage 3 of training for VLN-BERT and masks out 15% of the tokens. However as we see in Table 7.1 there is a low density of spatial words and directional words such as ‘left’ and ‘right’. In order to encourage the model to learn the connection between directional words and the path we add an additional training stage at the end of stage 3 where we mask out all spatial and directional words and train using only the loss on the language, and do not mask out any of the image tokens. We train only the model for 10 epochs with this objective. Then we fine-tune the model with the orginal cross-entropy loss and R2R training data. We report results of this model in Table 7.3 and refer to this in the table as the spatial language loss (SLL).

Additionally we experiment with creating additional hard negatives. AirBert [134] in-

troduced an additional shuffling training objective and found it was a effective way to teach the model to reason about temporal oder. In this objective, the order of the shuffling the image or caption tokens are shuffled to create additional hard negatives and then the model is trained via cross entropy to predict the non-shuffled instruction prediction pair. To bring this shuffling strategy into the VLN-BERT, we shuffle the order of the image tokens and then use these as negative paths during the fine-tuning stage which uses the path ranking object via cross entropy loss. We report results of this model in Table 7.3 and refer to this in the table as shuffling.

7.6 Results

Table 7.3 shows the results of the models trained with shuffling based negatives, the spatial language loss (SLL) and passive data in addition to the R2R train split. We find that adding shuffling increases model performance significantly, 3.1% in SR over the original model. Adding the SLL increase SR by an additional 1.62%. We find that training VLN-Bert on passive data in addition to R2R data increases performance.

Note that all models have been retrained from stage 2, using ViLBERT model weights for stage 1 and 2. We found that after retraining stage 3 and 4 of the VLN-Bert model according to training specifications outlined in [8] we were unable to replicate the same accuracy as reported in their paper and see a 3.36% drop in SR. In Table 7.2 we use the VLN-Bert model provided by [8] which achieves higher SR.

In addition to the result over the main validation sets, we seek to identify if any of these experiments have increased the model's ability to exploit spatial information. To determine this we re-run the masking experiments over the VLN-Bert models trained with R2R, additional passive data, data augmentation and additional training objectives. We show the results of these experiments in Table 7.4.

We find that in addition to increasing performance, the shuffling + SLL model shows larger reliance on spatial words than the original model. When directional words are

Table 7.3: Results of the VLN-Bert architecture trained with different data augmentation and training objectives. Note that all models have been retrained from stage 2, using ViLBERT model weights for stage 1 and 2.

		val-seen			val-unseen		
Method	Data	NE ↓	SR ↑	SPL ↑	NE ↓	SR ↑	SPL ↑
Original Loss	R2R	**4.1093**	**0.6667**	**0.6309**	4.6699	0.5590	0.5158
Shuffling	R2R	4.2836	0.6608	0.6209	4.5222	0.5900	0.5436
Shuffling + SLL	R2R	4.3506	0.6490	0.6105	**4.1503**	**0.6062**	**0.5608**
Original Loss	R2R + Passive Data	4.2104	0.6461	0.6061	4.2401	0.5607	0.5197

Table 7.4: Results of the token type masking experiments across different versions of the VLN-BERT model over the val-unseen split of R2R. The results are shown in terms of Success Rate (SR). SR of VLN models measures the percentage of selected paths that stop within 3m of the goal. The first column is the models performance with no augmentation to the input language.

	masking - criteria						
Model	No Masking	Swap	Left-Right	Spatial Words	Adjectives	Nouns	Verbs
VLN-Bert Original	0.5590	0.5607	0.5556	0.5564	0.5577	0.4368	0.5577
Shuffling	0.5900	0.5641	0.5824	0.5726	0.5866	0.4576	0.5896
Shuffling + SLL	0.6062	0.5743	0.6041	0.5900	0.6041	0.4334	0.5994
Add Passive Data	0.5607	0.5628	0.5641	0.5560	0.5607	0.3959	0.5513

swapped in the original model, performance increases slightly. In contrast the shuffling + SLL model suffers a drop in performance of 3.19%, which is the largest performance drop for the masking experiments outside of noun masking. Based on the results of these masking experiments we can assume the shuffling + SLL model is leveraging more spatial and directional information than the original model. We find that training with passive data and the original losses retains effect of reliance on nouns.

7.7 Conclusion

Approaches to the Vision Language Navigation task largely fall in two categories, discriminative and generative models. In this chapter we highlight a significant limitation of the discriminative VLN models. We propose a set of directed token masking experiments at

inference time over VLN models to deduce what parts of the text instructions the models attend to. Via these experiments we find that the most popular discriminative VLN models seem to solely rely on the nouns of the navigation instruction. In order to encourage the discriminative models to leverage other parts of the instruction we experiment with additional training objectives, data augmentation and adding additional training data and find some of these strategies to be effective and lead to higher performance on the VLN task.

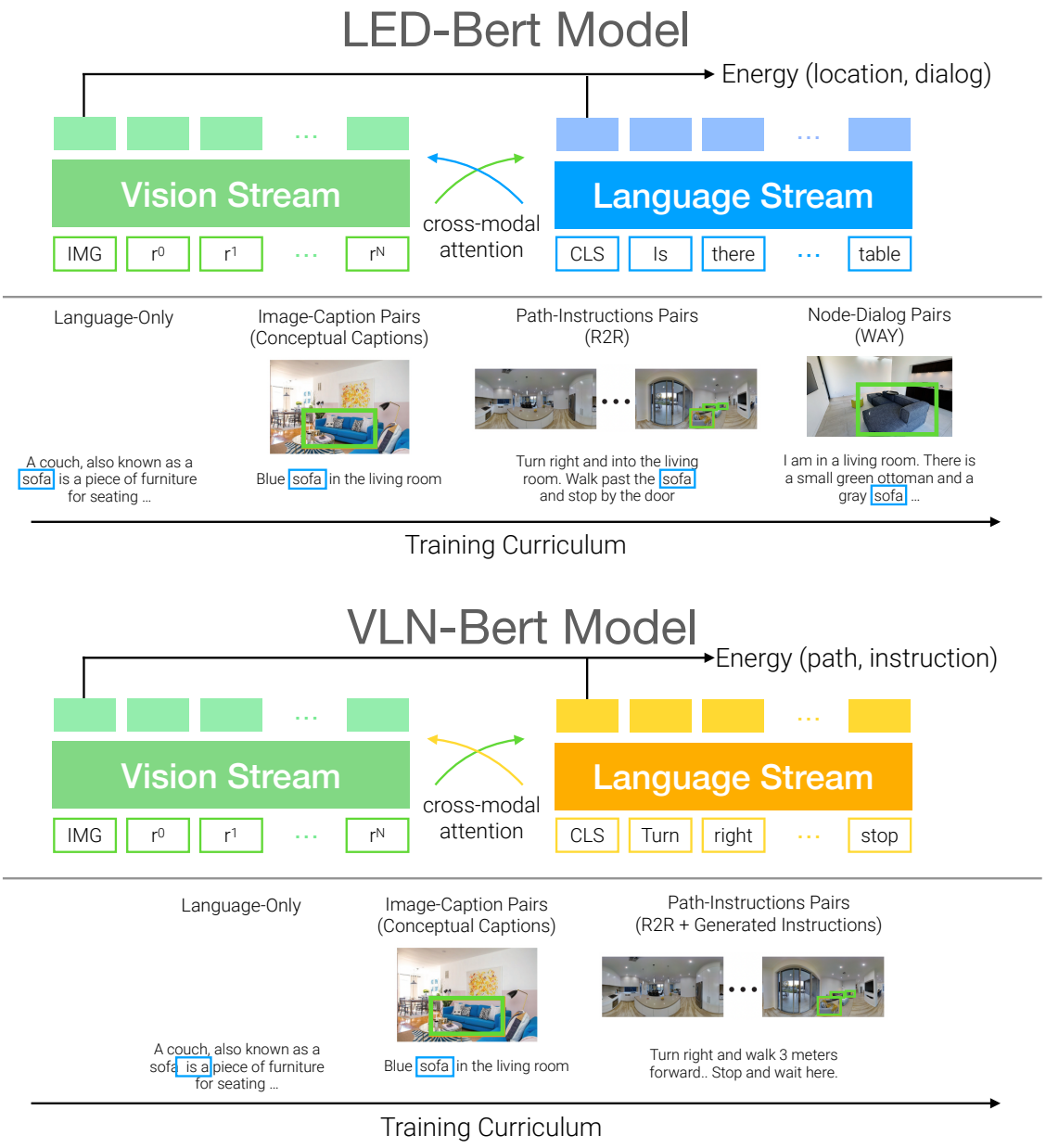

Figure 7.1: Illustration of the crossmodal BERT based models of LED-BERT and VLN-BERT and their training procedures.

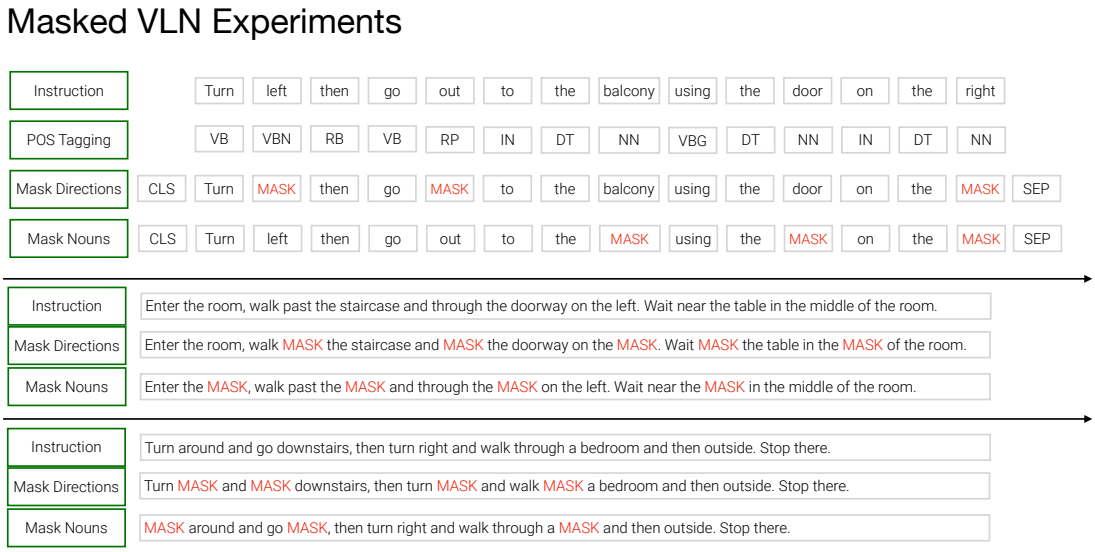

Figure 7.2: Example of how the navigational instructions of the R2R dataset are augmented during the masking test experiments. The instructions are part of speech tagged and tokenized. Different tokens are masked out depending on the experiment criterion. In the SWAP experiment 'left' and 'right' tokens are swapped and no token is masked.

CHAPTER 8
CONCLUSION

In this dissertation, we have examined several datasets, tasks and techniques, which situate agents in multi-modal environments and require them to either navigate through the environment or localize something within the environment. Within these tasks, these embodied agents are enabled with the ability to actively interact with other agents via language or their environment via actions. In the Temporal Activity Localization via Language Query (TALL) task (Chapter 3) we consider an agent that is navigating through a visual-temporal space similarly to how an agent navigates a visual 3D space and demonstrate this agent can be learned using reinforcement learning and gated-attention for language grounding. In the Image-Goal Navigation task (Chapter 4) we evolve this navigation agent to a embodied agent in a 3D space which has the constraints of the interaction and traversal cost of moving an a physical environment. This navigation task is simplified by only using the visual modality and we show a navigation agent can be learnt without reinforcement learning (RL) and instead using passive data to do distance prediction. In Localization Via Embodied Dialog (Chapter 5 and 6), we re-introduce the multi-modality to the embodied agents but remain in the 3D space. This task contains further complexity of localization, navigation and multiple agents. We find again that utilizing pre-training and passive data from the web and other tasks greatly improves our agents performance. And finally in Chapter 7, we examine how specific pre-training schemes can effect the behavior of multi-modal localization and navigation agents.

Largely this thesis focused on how to teach embodied agents to navigate and localize using dialog in unknown environments and in the face of limited scene information. We presented a new dataset and designed three dialog based localization tasks on top of the dataset. In this work we only explored modeling for one of the tasks. However the second

task Embodied Visual Dialog (EVD) is still unexplored and has a lot of potential as a novel task in the EAI space. EVD involves predicting the navigation actions of the observer at each time-step in the dialog. We believe it is the logical next step for dialog based localization tasks as one could transfer the LED-Bert and graph model to serve as baselines for the EVD task.

We believe our findings in Chapter 7 can serve as a indication for the need of direct analysis on EAI agents behavior to ensure the behavior is not due to biases in datasets. Based on our findings believe that the discriminative approach to the VLN task is reducing the complexity of the task such that the model can rely on biases of the dataset to achieve high performance while leveraging a small percentage of words in the instructions. To this end we suggest future work focus on increasing the complexity of the dataset and continue to use the analytical masking experiments we proposed in this dissertation.

REFERENCES

[1] M. Hahn, A. Kadav, J. M. Rehg, and H. P. Graf, "Tripping through time: Efficient localization of activities in videos," *BMVC*, 2019.

[2] M. Hahn, D. Chaplot, S. Tulsiani, M. Mukadam, J. Rehg, and A. Gupta, "No rl, no simulation: Learning to navigate without navigating," 2021.

[3] M. Hahn, J. Krantz, D. Batra, D. Parikh, J. M. Rehg, S. Lee, and P. Anderson, "Where are you? localization from embodied dialog," 2020.

[4] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018.

[5] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *CoRL*, 2019.

[6] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra, "Visual dialog," in *CVPR*, 2017.

[7] V. Murahari, D. Batra, D. Parikh, and A. Das, "Large-scale pretraining for visual dialog: A simple state-of-the-art baseline," in *European Conference on Computer Vision*, Springer, 2020.

[8] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra, "Improving vision-and-language navigation with image-text pairs from the web," in *ECCV*, 2020.

[9] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *CVPR*, 2018.

[10] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *CVPR*, 2018.

[11] H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi, "Touchdown: Natural language navigation and spatial reasoning in visual street environments," in *CVPR*, 2019.

[12] H. Mehta, Y. Artzi, J. Baldridge, E. Ie, and P. Mirowski, "Retouchdown: Adding touchdown to streetlearn as a shareable resource for language grounding tasks in street view," *arXiv preprint arXiv:2001.03671*, 2020.

[13] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. v. d. Hengel, "Reverie: Remote embodied visual referring expression in real indoor environments," in *CVPR*, 2020.

[14] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *AAAI*, 2018.

[15] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *CVPR*, 2018.

[16] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *CVPR*, 2020.

[17] K. Nguyen and H. Daumé III, "Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning," in *EMNLP*, 2019.

[18] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *3DV*, 2017.

[19] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *ICCV*, 2017.

[20] H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville, "Guesswhat?! visual object discovery through multi-modal dialogue," in *CVPR*, 2017.

[21] V. Blukis, D. Misra, R. A. Knepper, and Y. Artzi, "Mapping navigation instructions to continuous control actions with position-visitation prediction," in *CoRL*, 2018.

[22] P. Anderson, A. Shrivastava, D. Parikh, D. Batra, and S. Lee, "Chasing ghosts: Instruction following as bayesian state tracking," in *NeurIPS*, 2019.

[23] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi, "Learning to map natural language instructions to physical quadcopter control using simulated flight," in *CoRL*, 2019.

[24] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi, "Mapping instructions to actions in 3d environments with visual goal prediction," in *EMNLP*, 2018.

[25] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "Referitgame: Referring to objects in photographs of natural scenes," in *EMNLP*, 2014.

[26] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy, "Generation and comprehension of unambiguous object descriptions," in *CVPR*, 2016.

[27] J. Gao, C. Sun, Z. Yang, and R. Nevatia, "Tall: Temporal activity localization via language query," *ICCV*, 2017.

[28] L. A. Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell, "Localizing moments in video with natural language," in *ICCV*, 2017.

[29] J. Chen, X. Chen, L. Ma, Z. Jie, and T.-S. Chua, "Temporally grounding natural sentence in video," in *EMNLP*, 2018.

[30] M. Liu, X. Wang, L. Nie, X. He, B. Chen, and T.-S. Chua, "Attentive moment retrieval in videos," in *Conference on Research & Development in Information Retrieval*, 2018.

[31] X. Song and Y. Han, "Val: Visual-attention action localizer," in *Pacific Rim Conference on Multimedia*, 2018.

[32] Y. Yuan, T. Mei, and W. Zhu, "To find where you talk: Temporal sentence localization in video with attention based location regression," *AAAI*, 2019.

[33] H. Xu, K. He, L. Sigal, S. Sclaroff, and K. Saenko, "Text-to-clip video retrieval with early fusion and re-captioning," *arXiv preprint arXiv:1804.05113*, 2018.

[34] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[35] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Neurips*, 2015.

[36] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid, "Weakly-supervised alignment of video with text," in *ICCV*, 2015.

[37] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," in *CVPR*, 2018.

[38] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Qiu Chen, "Temporal context network for activity localization in videos," in *ICCV*, 2017.

[39] J. Yuan, B. Ni, X. Yang, and A. A. Kassim, "Temporal action localization with pyramid of score distribution features," in *CVPR*, 2016.

[40] H. Xu, A. Das, and K. Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," *ICCV*, 2017.

[41] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, "Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," in *CVPR*, 2017.

[42] C. Rodriguez-Opazo, E. Marrese-Taylor, F. S. Saleh, H. Li, and S. Gould, "Proposal-free temporal moment localization of a natural-language query in video using guided attention," in *WACV*, 2020.

[43] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *CVPR*, 2016.

[44] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. Peter Graf, "Attend and interact: Higher-order object interactions for video understanding," in *CVPR*, 2018.

[45] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.

[46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3d: Generic features for video analysis," *CoRR*, 2014.

[47] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *CVPR*, 2016.

[48] H. Alwassel, F. Caba Heilbron, and B. Ghanem, "Action search: Spotting actions in videos and its application to temporal action localization," in *ECCV*, 2018.

[49] D. He, X. Zhao, J. Huang, F. Li, X. Liu, and S. Wen, "Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos," *arXiv preprint arXiv:1901.06829*, 2019.

[50] W. Wang, Y. Huang, L. Wang, and, "Language-driven temporal activity localization: A semantic matching reinforcement learning model," in *CVPR*, 2019.

[51] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[52] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, 2017.

[53] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *CVPR*, 2017.

[54] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *ICLR*, 2020.

[55] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects," in *arXiv:2006.13171*, 2020.

[56] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, "Bayesian relational memory for semantic visual navigation," in *ICCV*, 2019.

[57] L. Mezghani, S. Sukhbaatar, T. Lavril, O. Maksymets, D. Batra, P. Bojanowski, and K. Alahari, "Memory-augmented reinforcement learning for image-goal navigation," *arXiv preprint arXiv:2101.05181*, 2021.

[58] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," *ICLR*, 2019.

[59] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *ICLR*, 2018.

[60] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Occupancy anticipation for efficient exploration and navigation," in *ECCV*, 2020.

[61] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, "Habitat: A platform for embodied ai research," in *ICCV*, 2019.

[62] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Neurips*, 2019.

[63] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to navigate in cities without a map," in *Neurips*, 2018.

[64] S. Emmons, A. Jain, M. Laskin, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse graphical memory for robust planning," in *Neurips*, 2020.

[65] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vázquez, and S. Savarese, "A behavioral approach to visual navigation with graph localization networks," in *Robotics: Science and Systems*, 2019.

[66] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," *ICRA*, 2020.

[67] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *ICLR*, 2018.

[68] D. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *CVPR*, 2020.

[69] M. Chang, A. Gupta, and S. Gupta, "Semantic visual navigation by watching youtube videos," in *Neurips*, 2020.

[70] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *ICLR*, 2020.

[71] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Neurips*, 2020.

[72] R. A. Epstein, E. Z. Patai, J. B. Julian, and H. J. Spiers, "The cognitive map in humans: Spatial navigation and beyond," *Nature Neuroscience*, 2017.

[73] E. C. Tolman, "Cognitive maps in rats and men.," *Psychological Review*, 1948.

[74] P. Foo, W. H. Warren, A. Duchon, and M. J. Tarr, "Do humans integrate routes into a cognitive map? map-versus landmark-based navigation of novel shortcuts.," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 2005.

[75] R. F. Wang and E. S. Spelke, "Human spatial representation: Insights from animals," *Trends in Cognitive Sciences*, 2002.

[76] H. Wang, W. Wang, W. Liang, C. Xiong, and J. Shen, "Structured scene memory for vision-language navigation," in *Conference on Computer Vision and Pattern Recognition*, 2021.

[77] B. Pan, H. Cai, D.-A. Huang, K.-H. Lee, A. Gaidon, E. Adeli, and J. C. Niebles, "Spatio-temporal graph for video captioning with knowledge distillation," in *CVPR*, 2020.

[78] X. Yang, K. Tang, H. Zhang, and J. Cai, "Auto-encoding scene graphs for image captioning," in *CVPR*, 2019.

[79] D. Teney, L. Liu, and A. van Den Hengel, "Graph-structured representations for visual question answering," in *Conference on Computer Vision and Pattern Recognition*, 2017.

[80] W. Norcliffe-Brown, E. Vafeias, and S. Parisot, "Learning conditioned graph structures for interpretable visual question answering," *arXiv preprint arXiv:1806.07243*, 2018.

[81] K. Chen, J. K. Chen, J. Chuang, M. Vázquez, and S. Savarese, "Topological planning with transformers for vision-and-language navigation," in *Conference on Computer Vision and Pattern Recognition*, 2021.

[82] Y. Hong, C. Rodriguez-Opazo, Y. Qi, Q. Wu, and S. Gould, "Language and visual entity relationship graph for agent navigation," *arXiv preprint arXiv:2010.09304*, 2020.

[83] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, 2018.

[84] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[85] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015.

[86] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[87] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi, "A corpus for reasoning about natural language grounded in photographs," *arXiv preprint arXiv:1811.00491*, 2018.

[88] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 201–216.

[89] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles, "Dense-captioning events in videos," in *ICCV*, 2017.

[90] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *CVPR*, 2012.

[91] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.

[92] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[93] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *Neurips*, 2014.

[94] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," *ICLR*, 2016.

[95] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *ICLR*, 2015.

[96] M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal, "Grounding action descriptions in videos," *ACL*, 2013.

[97] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments," in *Computer Vision and Pattern Recognition*, 2018.

[98] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *CVPR*, 2017.

[99] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[100] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *ICLR*, 2017.

[101] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, 2007.

[102] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[103] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," in *SIGGRAPH*, 2018.

[104] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *CVPR*, 2018.

[105] W. Falcon and H. Schulzrinne, "Predicting floor-level for 911 calls with neural networks and smartphone sensor data," in *ICLR*, 2018.

[106] J. Baldridge, T. Bedrax-Weiss, D. Luong, S. Narayanan, B. Pang, F. Pereira, R. Soricut, M. Tseng, and Y. Zhang, "Points, paths, and playscapes: Large-scale spatial language understanding tasks set in the real world," in *International Workshop on Spatial Language Understanding*, 2018.

[107] J. L. McClelland, F. Hill, M. Rudolph, J. Baldridge, and H. Schütze, "Extending machine language models toward human-level language understanding," *arXiv preprint arXiv:1912.05877*, 2019.

[108] Y. Bisk, A. Holtzman, J. Thomason, J. Andreas, Y. Bengio, J. Chai, M. Lapata, A. Lazaridou, J. May, A. Nisnevich, N. Pinto, and J. Turian, "Experience grounds language," *arXiv preprint arXiv:2004.10151*, 2020.

[109] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *EMNLP*, 2016.

[110] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela, "Talk the walk: Navigating new york city through grounded dialogue," *arXiv preprint arXiv:1807.03367*, 2018.

[111] E. Loper and S. Bird, "Nltk: The natural language toolkit," *arXiv preprint arXiv:0205028*, 2002.

[112] M. Kozhevnikov, M. A. Motes, B. Rasch, and O. Blajenkova, "Perspective-taking vs. mental rotation transformations and how they predict spatial navigation performance," *Applied Cognitive Psychology*, 2006.

[113] G. Ilharco, V. Jain, A. Ku, E. Ie, and J. Baldridge, "General evaluation for instruction conditioned navigation using dynamic time warping," *arXiv preprint arXiv:1907.05446*, 2019.

[114] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[115] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[116] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[117] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[118] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.

[119] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *ACL*, 2002.

[120] M. Hahn, J. Krantz, D. Batra, D. Parikh, J. M. Rehg, S. Lee, and P. Anderson, "Where are you? localization from embodied dialog," *EMNLP*, 2020.

[121] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[122] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[123] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[124] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[125] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," *arXiv preprint arXiv:1908.07490*, 2019.

[126] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and vqa," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[127] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[128] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[129] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[130] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: A comprehensive review," *Computational Social Networks*, 2019.

[131] D. Li, Q. Zhang, D. Zhao, Y. Zhuang, B. Wang, W. Liu, R. Tutunov, and J. Wang, "Graph attention memory for visual navigation," *arXiv preprint arXiv:1905.13315*, 2019.

[132] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding," *arXiv preprint arXiv:2010.07954*, 2020.

[133] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. v. d. Hengel, "Reverie: Remote embodied visual referring expression in real indoor environments," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9982–9991.

[134] P.-L. Guhur, M. Tapaswi, S. Chen, I. Laptev, and C. Schmid, "Airbert: In-domain pretraining for vision-and-language navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1634–1643.

[135] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[136] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "Vln bert: A recurrent vision-and-language bert for navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1643–1653.

[137] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," *Advances in Neural Information Processing Systems*, 2018.

[138] W. Hao, C. Li, X. Li, L. Carin, and J. Gao, "Towards learning a generic agent for vision-and-language navigation via pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 137–13 146.

[139] S. Wang, C. Montgomery, J. Orbay, V. Birodkar, A. Faust, I. Gur, N. Jaques, A. Waters, J. Baldridge, and P. Anderson, "Less is more: Generating grounded navigation instructions from landmarks," *arXiv preprint arXiv:2111.12872*, 2021.

[140] M. Zhao, P. Anderson, V. Jain, S. Wang, A. Ku, J. Baldridge, and E. Ie, "On the evaluation of vision-and-language navigation instructions," *arXiv preprint arXiv:2101.10504*, 2021.

[141] W. Zhu, Y. Qi, P. Narayana, K. Sone, S. Basu, X. E. Wang, Q. Wu, M. Eckstein, and W. Y. Wang, "Diagnosing vision-and-language navigation: What really matters," *arXiv preprint arXiv:2103.16561*, 2021.

[142] Y. Zhang, H. Tan, and M. Bansal, "Diagnosing the environment bias in vision-and-language navigation," *arXiv preprint arXiv:2005.03086*, 2020.

[143] J. Thomason, D. Gordon, and Y. Bisk, "Shifting the baseline: Single modality performance on visual navigation & qa," *arXiv preprint arXiv:1811.00613*, 2018.