General Vision and Language Methods in Real Applications: A Focus on Vision-and-Language Navigation

Yanyuan Qiao

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

July 1, 2023

Contents

List of Figures

List of Tables

University of Adelaide

Abstract

General Vision and Language Methods in Real Applications: A Focus on Vision-and-Language Navigation

by Yanyuan Qiao

The field of Vision and Language has aroused significant interest and holds tremendous potential for real applications, particularly in the area of Vision-and-Language Navigation (VLN). The VLN task enables robots to understand navigation instructions expressed in natural language, perceive the environment, and execute corresponding actions, making it applicable in various scenarios such as home assistants. Despite considerable progress in advancing the development of VLN, several challenges persist and warrant further attention. These challenges include the lack of pre-training models that emphasize temporal information specific to VLN, the necessity for parameter-efficient transfer learning techniques to effectively utilize pre-training models, and the exploration of Large Language Models (LLMs) to leverage their extensive knowledge for enhanced performance in VLN. In this thesis, we propose a series of new methods to address these challenges. First, we introduce a history-enhanced and order-aware pre-training and fine-tuning paradigm for VLN. We design three VLN-specific proxy tasks: Action Prediction with History (APH) task, Trajectory Order Modeling (TOM) task and Group Order Modeling (GOM) task. Furthermore, we develop a memory network to address the representation inconsistency of history context between the pre-training and the fine-tuning stages. Second, we propose the first study exploring Parameter-Efficient Transfer Learning (PETL) methods for VLN tasks and propose a VLN-specific PETL method named VLN-PETL. Specifically, we design two PETL modules: Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB), which are integrated with existing PETL methods such as Adapter and LoRA to form the comprehensive VLN-PETL framework. Finally, we present a March-in-Chat (MiC) model, enabling conversations between the REVERIE agent and an LLM proactive planning of future steps. This model contains three modules: Goal-Oriented Static Planning (GOSiP) module, Scene-Oriented Dynamic Planning (SODiP) module, and one Room-and-Object Aware Scene Perceiver (ROASeP) module. Through Extensive quantitative and qualitative experiments, we demonstrate the efficiency and potential of our contributions to advancing the field of VLN.

Declaration of Authorship

Yanyuan Qiao

July 1, 2023

Acknowledgements

The journey of pursuing my doctoral degree at the University of Adelaide has been filled with countless cherished memories. This period of academic pursuit has not only expanded my knowledge and research horizons but has also provided me with valuable experiences that have shaped my career aspirations. I am deeply grateful for the support and guidance I have received from numerous individuals who have played a pivotal role in my academic journey.

First of all, I would like to express my heartfelt appreciation to my Ph.D. supervisor, Associate Professor Qi Wu, for all his exceptional guidance, unwavering support, and invaluable mentorship throughout my study. His expertise, patience, and encouragement have been instrumental in shaping my research and academic growth.

I am also grateful to my Co-supervisor, Dr.Yuankai Qi, for his support in my Ph.D. study. His expertise has greatly enriched my research and broadened my perspectives.

I would like to thank Prof. Peng Wang and Prof. Jing Liu. They have helped me a lot during my Ph.D. career with my research.

I am thankful to all my lab mates and friends, including but not limited to Yutong Xie, Yicong Hong, Xinyu Wang, Chaorui Deng, Qi Chen, Chongyang Zhao, Chaohan Wang, Gengze Zhou, Zheng Yu, Feng Chen, Ziqin Zhou, Yusi Feng, Qiaoyang Luo, Yutong Dai, Yangyang Shu, Yang Zhao, Na Sai, Fengyi Yang, Jinchao Ge, Xuan Ren. Their insightful discussions, collaborative spirit, and constructive feedback have greatly enriched my understanding and contributed to the development of this work. Their camaraderie and friendship have made the research environment stimulating and enjoyable.

In addition, I am thankful for the assistance and support provided by the technical and administrative staff at Australian Institute for Machine Learning (AIML). Their dedication and professionalism have greatly facilitated my research endeavors and administrative processes, ensuring a smooth academic experience.

Finally, I extend my sincere appreciation to my family for their unwavering love, understanding, and support throughout my academic journey. Their encouragement and belief in me have been a constant source of motivation, inspiring me to strive for excellence.

Publications

This thesis contains the following works that have been published or prepared for publication:

- HOP: History-and-Order Aware Pre-training for Vision-and-Language Navigation.
  Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, Qi Wu.
  Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

- HOP+: History-enhanced and Order-aware Pre-training for Vision-and-Language Navigation.
  Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, Qi Wu, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2023.

- VLN-PETL: Parameter-Efficient Transfer Learning for Vision-and-Language Navigation.
  Yanyuan Qiao, Zheng Yu, Qi Wu.
  International Conference on Computer Vision (ICCV), 2023.

- March in Chat: Interactive Prompting for Remote Embodied Referring Expression.
  Yanyuan Qiao, Yuankai Qi, Zheng Yu, Jing Liu, Qi Wu.
  International Conference on Computer Vision (ICCV), 2023.

In addition, I have the following papers not included in this thesis:

- Referring Expression Comprehension: A Survey of Methods and Datasets.
  Yanyuan Qiao, Chaorui Deng, Qi Wu.
  IEEE Transactions on Multimedia (TMM), 2020.

- R-GAN: Exploring Human-like Way for Reasonable Text-to-Image Synthesis via Generative Adversarial Networks.
  Yanyuan Qiao, Qi Chen, Chaorui Deng, Ning Ding, Yuankai Qi, Mingkui Tan, Xincheng Ren, Qi Wu.
  ACM Multimedia (ACM MM), 2021.

Chapter 1

Introduction

Vision-and-Language plays a vital role in our daily life and aroused huge interest in deep learning research [124, 72]. The synergy between vision and language brings great hope for advancing the frontier of artificial intelligence and creating intelligent agents, which can cooperate in a wide range of tasks and applications to assist human life [114]. The development of vision and language could bring various real applications, such as Visual Question Answering [6], Image Captioning [4], Referring Expression [128], Text-to-Image Synthesis [100, 137], especially in the research of Vision-and-Language Navigation (VLN) [28, 5]. Vision-and-Language Navigation task aims at allowing the robot to understand navigation instructions expressed in natural language and to perform related actions.

In this chapter, we first provide a comprehensive overview of the VLN task, including its problem definition and an overview of related works. Subsequently, we delve into the motivation of this thesis and identify existing challenges for the VLN task. In light of these challenges, we present our contributions and provide an outline of the thesis structure.

1.1 Background

In the Vision-and-Language Navigation task, an agent is required to follow natural language instructions given by the human (oracle) and navigate to reach the goal destination or find the target object. The agent first receives natural language instructions and obtains observations from the environment, and then takes actions to interact with the environment and get rewards that reflect the underlying task. The process of navigation of the agent is shown in Figure 1.1.

To be specific, the agent is required to predict the next step action based on natural language instruction and current visual observation. The environment is an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{V_i\}_{i=1}^{K}$ denotes K navigable nodes, and $\mathcal{E}$ denotes connectivity edges. At first, the agent is positioned with the initial state s_0 and given a natural language instruction $X = \langle x_1, x_2, \dots x_L \rangle$, where L is the instruction length, containing a sequence of L words. At each time step t, the agent perceives a panoramic view as the visual observation. The panoramic view contains 36 single

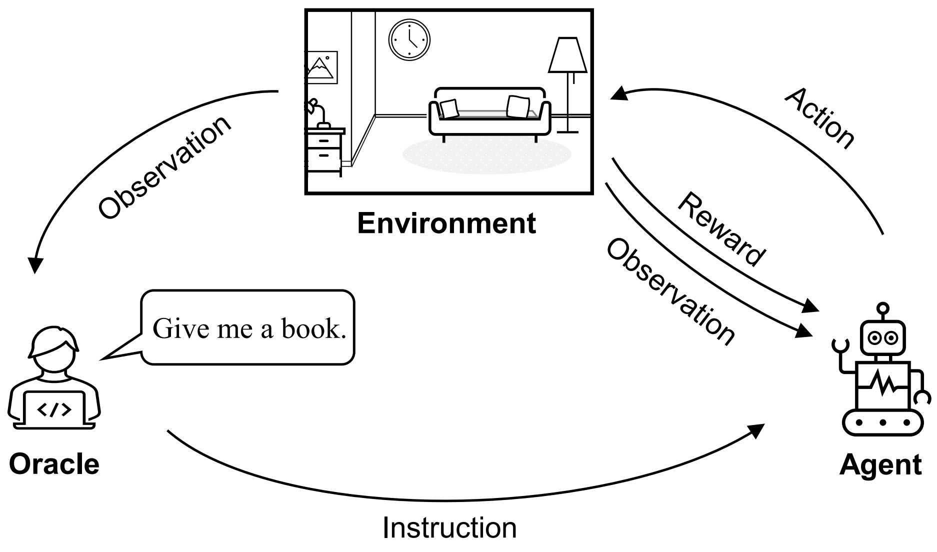


FIGURE 1.1. Illustration of Vision-and-Language Navigation Task.

views from 12 surrounding angles, each with 3 camera poses (up, down, horizon). Each view representation is the concatenation of its visual and orientation representation feature. Then the agent predicts the next action by selecting a navigable node from the candidate list. The agent navigates the environment sequentially and generates a trajectory $\tau = \langle v_1, v_2, \dots v_T \rangle$ of length T. The agent navigates in the environment until the special `[STOP]` action is selected, or the agent reaches a pre-defined maximum trajectory length.

In the past few years, VLN has received great attention and a large number of methods have been proposed to handle VLN [39, 38, 93, 67, 65] task. Early works were based on encoder-decoder frameworks [5, 76, 24]. While subsequent works approached VLN research in a variety of ways, such as data augmentation [24, 69, 59] to improve the robustness of the agent, progress monitoring [76, 120, 136] to estimate the completeness of instruction-following, and back-tracking [77, 71] to help the agent learn to decide when to perform backtracking depending on the state of the agent, *etc.* Recently, BERT-based pre-training methods significantly improved agents' performance on VLN tasks [61, 29]. These methods follow the pretrain-and-finetune paradigm to solve VLN tasks, which pretrain a VLN model on large instruction-and-trajectory data with specifically designed proxy tasks and finetune a full model for each downstream VLN task. VLNBERT [80] first proposes the pretraining-and-finetuning paradigm to solve different downstream VLN tasks, which introduces an extra proxy task of scoring path-instruction pairs in addition to the Masked Language Modeling proxy task. By learning the matching degree between the instructions and the panoramic image sequence, the agent will select the most appropriate path according to the instructions. PREVALENT [30] introduces a single-step action prediction proxy task, aiming to learn action-oriented generic visio-linguistic representation. HAMT [15] encodes past panoramic observations as historical information explicitly.

1.2 Motivation

While there have been numerous works that have made contributions to advancing the development of the VLN task, however, there still remain several challenges that need to be addressed:

- One of the main challenges is the lack of a pre-trained model that emphasizes temporal information specifically designed for the VLN task. Since VLN is a Partially Observable Markov Decision Process (POMDP), where the agent relies heavily on historical experiences to make the next action decision. In addition, VLN is a spatiotemporal task that is sensitive to the sequence order of the trajectory. Thus the ability of temporal order reasoning is also beneficial to action decision-making. Nevertheless, existing methods do not explicitly capture temporal order information from either instructions or visual observations.

- When using the pre-trained models, though these pretraining-and-finetuning methods have attempted to utilize vital historical knowledge for action prediction, the exploration is still limited due to the gap between the pretraining and fine-tuning stages. Meanwhile, these methods face the same challenging during the fine-tuning stage: oversized parameters for training and storage. Specifically, all parameters of the full pre-trained model will be trained and stored for each downstream VLN task. This would hinder the application of VLN in real-world scenarios since realistic robots will have difficulty in training and storing such huge parameters for every new task. It is essential to study parameter-efficient transfer learning (PETL) for VLN tasks.

- The emergence of Large Language Models (LLMs) has brought significant potential for research on VLN, since LLMs contain rich common sense knowledge, allowing for the acquisition of deeper information through prompt learning rather than relying on fine-tuning. In addition, navigating through high-level instructions poses another formidable task in the VLN domain. For instance, the instructions of REVERIE are closer to what we would say to an intelligent domestic robot in daily life in terms of the instruction length and logic, which is usually short and concise. However, most existing methods are primarily designed for VLN tasks involving detailed step-by-step instructions, thus they do not perform well on REVERIE. Consequently, it is crucial to devise novel approaches for leveraging the wealth of knowledge provided by LLMs and effectively applying it to VLN tasks.

1.3 Contribution and Thesis Outline

Motivated by the aforementioned challenges, the main focus of this thesis is to investigate the methods for effectively addressing these issues in the VLN task. The main contributions are summarized below:

- We propose a History-Enhanced and Order-Aware Pre-training with the complementing fine-tuning paradigm to solve the issues existing in the previous Vision-and-Language Navigation pre-training methods. We first carefully examine and compare previous methods, and we find these methods either overlook the important historical context in pre-training or neglect the role of the action order. Thus, we design an Action Prediction with History (APH) task that provides history visual observations of the action prediction in the pre-training. We then propose two order-aware proxy tasks, including Trajectory Order Modeling (TOM) and Group Order Modeling (GOM). In addition, we design an external memory network in fine-tuning stage to utilize the historical information to assist action prediction.
- We introduce the first investigation into the utilization of Parameter-Efficient Transfer Learning (PETL) methods to VLN tasks. We propose a VLN-specific PETL approach that introduces two PETL modules, namely Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). These modules are designed to enable efficient tuning of large pre-trained models for VLN downstream tasks. Furthermore, we incorporate the vanilla adapters to efficiently tune the language encoder and employ LoRA to further enhance the overall performance.
- We present a novel model March-in-Chat (MiC), which is specifically tailored for the REVERIE task, where the VLN agent receives brief high-level instructions. MiC empowers the REVERIE agent to engage in real-time conversations with a Large Language Model (LLM) to generate plans for upcoming steps. The model comprises three core modules: Goal-Oriented Static Planning (GOSiP), Scene-Oriented Dynamic Planning (SODiP), and Room-and-Object Aware Scene Perceiver (ROASeP) module. These modules work collaboratively to facilitate effective navigation.

Based on the aforementioned contributions, we organize the thesis structure as follows:

Chapter 2 provides a comprehensive review of the background in Vision and Language, Embodied AI, and the Vision-and-Language Navigation task. It covers essential aspects such as simulators, datasets, and current methodologies employed in VLN tasks.

Chapter 3 introduces a pre-training and fine-tuning paradigm with VLN-specific objectives. This paradigm utilizes past observations and concentrates on temporal information to support future action prediction, offering a unique perspective on enhancing VLN performance.

Chapter 4 focuses on the exploration of Parameter-Efficient Transfer Learning (PETL) techniques for VLN tasks. This chapter represents the first investigation into PETL methods specifically tailored for VLN and proposes an innovative VLN-PETL approach.

Chapter 5 delves into the utilization of Large Language Models (LLMs) for VLN. The chapter presents a March-in-Chat (MiC) model that engages in conversations with LLMs, dynamically planning actions based on the newly developed Room-and-Object Aware Scene Perceiver (ROASeP).

Chapter 6 concludes by summarizing the primary contributions of this thesis and providing a discussion on the future directions of VLN tasks. It highlights potential areas of further research and explores the possibilities for advancing the field of VLN.

Chapter 2

Literature Review

In this chapter, we first review the development of Vision and Language. Then we introduce the concept of Embodied AI, highlighting its significance and unique characteristics. Finally, we delve into one of the major tasks in Embodied AI: Vision-and-Language Navigation (VLN) task and provide a detailed overview of VLN datasets and methods.

2.1 Vision and Language

Vision and Language is the research area focused on the intersection of computer vision and natural language processing, which aims to bridge the gap between visual and textual modality. The research of vision and language has seen remarkable growth and advancements in recent years. Early works focused on single tasks such as Image Captioning [4, 26] (generate text description from image), text-to-image generation, and Visual Question answering [6, 126](answer questions based on visual input), Visual Commonsense Reasoning (requires the model to use commonsense knowledge to reason about images).

Recently, significant progress has been made in the area of visual and language research due to advances in deep learning and the availability of large-scale multimodal datasets. Models like UNITER (UNiversal Image-TExt Representation) [18], Oscar (Object-Semantics Aligned Pre-training) [62], LXMERT (Cross-Modality Transformer) [115], and ViLBERT (Vision-Language BERT) [74] have demonstrated significant improvements in capturing visual-language correlations and achieving state-of-the-art performance across various tasks. CLIP (Contrastive Language-Image Pre-training) [97] is a powerful pre-trained model that learns joint representations of images and text, which is trained on a large corpus of image-text pairs using a contrastive loss. It can perform tasks like zero-shot image classification tasks.

Visual perception networks play an important role in enabling the model to understand visual content, analyzing its semantic information, and extracting visual features for multimodal understanding. ResNet [33] is a widely used deep convolutional neural network architecture for visual perception tasks. It is composed of multiple residual blocks with skip connections, which allows training very deep networks.

By introducing shortcut connections that bypass certain layers, ResNet allows the network to learn residual mappings, focusing on learning the difference between the input and the required output. These skip connections promote the flow of gradients and alleviate the degradation problem associated with deeper networks. ResNet variants (such as ResNet-50, ResNet-101, and ResNet-152) have different network depths, with more layers typically resulting in higher accuracy but increased computational cost. These architectures have become widely adopted and serve as the backbone for numerous computer vision tasks. They are often used for feature extraction, where the learned representations from pre-trained ResNet models are employed in transfer learning settings. ResNet models can effectively capture hierarchical visual features and have been successfully applied in various tasks such as image classification, object detection, and image understanding. In the VLN task, ResNet-152 pre-trained on Place365 [134] is usually used to extract visual features. Bottom-Up visual attention models, such as Faster-RCNN [102] or Mask R-CNN [32], focus on detecting and localizing objects within images. These models utilize Region Proposal Network (RPN) to generate region proposals and convolutional neural networks to extract features from these regions of interest. Bottom-up models are suitable for tasks that require object recognition. In the REVERIE task, the VLN model uses the Faster-RCNN pre-trained on the Visual-Genome [53] to encode object features. Vision Transformer (ViT) [23] extends the Transformer architecture originally designed for natural language processing to the computer vision domain. The ViT model divides an image into blocks and processes them with a Transformer encoder to capture global relations and contextual information. The ViT model has shown good performance in image classification tasks. For the VLN task, HAMT [15] uses ViT-B/16 for image encoding, which brings significant performance improvements.

Language understanding involves the comprehension and interpretation of natural language instructions, requiring the agent to understand the semantics, syntax, and contextual cues embedded in the textual instruction. This understanding is crucial for effectively navigating and interacting with visual content in environments. Early works [5] normally used LSTM (Long Short-Term Memory) networks [19] were commonly used to extract textual features. Subsequently, the development of transformer-based models shows superiority in language understanding and has been widely applied in VLN methods [39, 15].

The research of Vision-and-Language has made significant progress in integrating visual and language modalities to develop artificial intelligence capable of understanding and generating multimodal information. The field has evolved from individual tasks to more complex and interactive scenarios, pushing the boundaries of multimodal understanding and reasoning. With continuous advances in deep learning, pre-training and fine-tuning paradigm, and multimodal architectures, Vision and Language research holds tremendous potential for various applications, the Vision and Language research holds great potential for applications ranging from content generation to human-machine collaboration.

2.2 Embodied AI

Embodied Artificial Intelligence (Embodied AI) has emerged as a fascinating field at the intersection of artificial intelligence, robotics, and cognitive science [20]. It focuses on developing intelligent agents capable of perceiving, understanding, and interacting within real-world environments in a human-like manner. Unlike traditional AI approaches that mainly rely on static data from the internet and pre-defined rules, Embodied AI aims to create agents that can actively explore and navigate their surroundings, allowing them to acquire knowledge through direct interaction. In addition, Embodied AI agents could be physically located in the real world, interacting with their environment through sensory input such as vision and language.

The concept of Embodied AI dates back to the early days of artificial intelligence research. Researchers recognize the limitations of traditional AI approaches that rely solely on symbolic reasoning and static knowledge representation. Brooks *et al.* [8] introduce a behavior-based architecture for robots, emphasizing the importance of interaction with the physical world for intelligence, which is considered the foundation work in the field of Embodied AI. This marked a shift from traditional AI approaches focused on symbolic reasoning to a more embodied and interactive approach in AI research.

By combining visual perception with natural language understanding and applying it to Embodied AI, agents can perceive, understand and interact with the environment in a more human-like way, which can be applied to smart homes, self-driving cars, and other fields. Embodied AI empowers the agent to not only passively observe their surrounding environments but also actively navigate, communicate, and even manipulate objects to achieve specific goals.

Since training and testing agents directly in physical environments are expensive, poorly reproducible, and time-consuming, virtual environments simulate real-world scenarios, allowing researchers to test and refine their algorithms and architectures in a controlled environment. These simulators provide realistic physics and diverse environments, enabling researchers to generate large-scale training data for agent development. They also provide tools for benchmarking agent performance, promoting reproducibility and fair comparison across different methods. Simulators enable agents to interact with 3D scenes, receive sensory input, and perform actions, facilitating the training and evaluation of embodied AI models at scale. Mainstream simulators in the field of Embodied AI include Matterport [10, 5], AI2-THOR [51], Habitat [104], iGibson [106], and RoboTHOR [21], *etc.*

Embodied AI covers a range of tasks that require agents to perceive, reason, plan, and act within their environments. These tasks include Locomotion [56, 90, 125], Visual Navigation [82, 99], Visual-and-Language Navigation [5, 107, 118], Object

Manipulation [1, 44, 85] and Rearrangement [113], *etc.* Each task presents unique challenges and requires agents to integrate visual perception, language understanding, and physical actions to achieve their objectives.

2.3 Vision-and-Language Navigation

In recent years, the progress of Embodied AI has been accelerated by the development of vision and language, the availability of large-scale datasets, powerful deep-learning models, and advanced simulators. Vision-and-Language Navigation (VLN) [52] emerged as a key task within embodied AI, focusing on integrating vision and language understanding to enable agents to navigate and comprehend complex instructions in their environment. VLN tasks typically involve agents receiving natural language instructions and using visual perception to navigate through a scene toward a specific goal location. In the VLN task, the agent is required to develop a comprehensive understanding of their surroundings, interpret natural language instructions, and execute actions accordingly. And it has huge potential to apply for autonomous robots, virtual assistants, and augmented reality applications.

2.3.1 Simulator

Recent advancements in the field of VLN have witnessed the emergence of several prominent simulators, such as Matterport3D [10], Habitat [104, 114], and AI2-THOR [51]. These simulators offer researchers a controlled and realistic environment to study and develop VLN algorithms.

Matterport3D

Matterport3D Simulator [5] constructed based on the Matterport3D Dataset [10], which contains 10,800 panoramic views of 90 building-scale scenes. It includes diverse scenes such as houses, apartments, hotels, and offices. And the scenes are meticulously captured using 3D cameras, resulting in highly detailed and realistic reproductions of real-world spaces. It supports simulated RGB and depth cameras as well as semantic segmentation, allowing agents to extract visual information from their surrounding environments. Agents can navigate through complex spaces, through corridors, up and down the stairs, and between different rooms. The simulator provides local and global path-planning algorithms that enable agents to efficiently navigate the environment while taking obstacles and constraints into account.

Habitat

Habitat 1.0 [104] is developed by Facebook AI Research, and it provides a realistic and interactive virtual environment for training and evaluating agents in navigation

TABLE 2.1. Major datasets for VLN and their main characteristics.

Dataset	#Path	#Instr	#Words	Instr Length	Language
R2R [5]	7K	21.7K	625K	29	1
REVERIE [95]	7K	21.7K	388K	18	1
CVDN [117]	7K	2.1K†	167K	-	1
RxR [54]	16.5K	126.1K	9.8M	78	3

†The number of dialogues.

and interaction tasks. Habitat offers a diverse collection of highly detailed 3D environments that simulate real-world indoor scenes. The scenes are generated from real-world scans and exhibit detailed geometry, texture, and lighting. Habitat provides agents with realistic sensory inputs such as visual observations, depth maps, and semantic segmentation. This enables agents to perceive their surroundings and make informed decisions based on the available sensory information. Habitat 2.0 [114] is an upgraded version of Habitat 1.0, which is for agents in interactive 3D environments and supports piece-wise rigid objects as well as rigid-body mechanics. Habitat 2.0 supports a variety of actions and control mechanisms, enabling the agent to perform tasks such as navigation, object manipulation, and interaction with virtual objects as a home assistant.

AI2-THOR

AI2-THOR [51] is a highly interactive and realistic 3D simulator developed by Allen Institute for AI. The simulator provides diverse and dynamic indoor environments such as residential apartments and offices. It supports various sensory inputs, including RGB, depth, and semantic segmentation. AI2-THOR emphasizes interaction with objects and incorporates a realistic physics engine that enables agents to interact with objects in the environment. The agent can perform actions such as opening doors, picking up objects, and manipulating objects, thus facilitating realistic dynamic interactions between the agent and its surrounding environment.

2.3.2 Datasets

A number of datasets have been proposed specifically for the VLN task. In the following paragraphs, we provide an overview of the available datasets. Key characteristics are summarized in Table 2.1.

Room-to-Room (R2R)

Room-to-Room task [5] requires agents to follow detailed instructions to navigate from one room to another. These instructions contain rich linguistic information, such as “Exit the room. Walk across the hallway. Turn slightly right and walk across the kitchen towards the sofas. Turn left and walkthrough doorway just right of the portraits of a family.", "Go into the archway to the left of the room with the dining table into the room with the circle table in the middle, make a right towards the front door, take two steps up the stairs onto the level and stop.".

The R2R dataset is collected from the Matterport3D simulator [11]. The trajectory is the shortest path sampled from the start pose and goal location pairs in different rooms. It contains 21,567 navigation instructions, 7,189 trajectories, and 10,800 panoramic views of 90 real-world building-scale indoor environments. The average length of each instruction is 29 words. The R2R dataset consists of four splits: train, validation seen and validation unseen, and test unseen.

Room-Across-Room (RxR)

The task of RxR [54] is an updated version of R2R and is more challenging than R2R. For example, instructions in RxR are longer and more detailed, describing more landmarks and including dense spatiotemporal grounding than R2R does. Besides, instructions in RxR no longer describe the shortest path between the starting room to the ending room and the length variance of paths is very large. Thus, agents cannot simply go directly to the targets and cannot simply use the strong prior of path length to navigate.

The RxR dataset contains 126,069 navigation instructions and 16,522 trajectories. IR consists of training, validation, and test set. In the training set it contains 11,089 paths. The validation set contains 1,232 paths for val-seen and 1,517 paths for val-unseen. And test set has 2,684 paths. The average length of each instruction is 78 words. The instructions of RxR are in three languages (*i.e.* English, Hindi, and Telugu). Here are examples of instructions, "Okay, now you are in a room facing towards two bathtubs, one on the right side and the other on the left side. Now turn to your left and slightly move forward. Now slightly turn to your right and go straight and stand next to the white bathtub, which is on the left side. Now in front of you there are two steps, go straight and stand on the second step. Now you are standing on the second step with white bathtub on the left side and this is the end point", "You are facing towards the wall, turn left and move forward. You can see an open door right in front of you, move towards the door. Turn left and enter into the room. Turn right you can see sofa in-front of you, move towards the sofa. You are standing next to the sofa, which is your final destination. "

REVERIE

The task of Remote Embodied Visual referring Expressions in Real 3D Indoor Environments (REVERIE) [95] gives concise, high-level instructions referring to a remote object, such as "Close the kitchen window", "Go to the entryway and turn off the lamp". REVERIE requires the agent to follow instructions to navigate and identify the target object in the unseen environment.

The REVERIE dataset contains 21,702 instructions. The average length of each instruction is 18 words. The dataset has 10,567 panoramas and 4,140 target objects, divided into 489 categories. On average, each target viewpoint has 7 objects with 50 bounding boxes. REVERIE follows the same train/validation/test split strategy as the R2R dataset. The training set contains 59 scenes with over 2,353 objects and 10,466 instructions. The validation set contains 63 scenes, 953 objects, and 4,944 instructions. The test set contains 16 scenes, 834 objects, and 6,292 instructions.

Vision-and-Dialog Navigation (CVDN)

The CVDN dataset [117] is used for the NDH task, which contains 2050 human-human navigation dialog and over 7K trajectories. In the NDH task, the agent is required to find the target location based on the dialog history, which consists of multiple question-and-answer interactions between the agent and its partners. Unlike traditional VLN datasets, the CVDN dataset introduces dialogue information, simulating interactive dialogue between two humans. Dialogues consist of question-and-answer information, allowing agents to request additional information and interact cooperatively. It is much more challenging because the instructions from the dialog history are often ambiguous and unspecified. As a result, agents can hardly navigate to the final location directly. NDH has three settings: (1) Oracle, which utilizes the shortest path as ground truth observed by the Oracle; (2) Navigator, which uses the path adopted by the human navigator as ground truth; (3) Mixed, which takes the shortest path or the path of human if the human visits the target location. Here are examples of instructions, “navigator: Should I go up the stairs or toward the toilet? oracle: yes i belive so. navigator: Can you be more specific. Up the stairs or move toward the toilet? oracle: don’t go up the stairs look to your right and it’s not in the bathroom find a couch. navigator: Is that the couch I am looking for? oracle: it is blue in color has many pillows on it.”

2.3.3 VLN Methods

Letting an agent navigate in a simulated environment according to instructions has attracted increasing attention, and there have been many studies on the Vision-and-Language Navigation task. Early works were based on an encoder-decoder framework [5], which utilizes an LSTM-based sequence-to-sequence architecture with an attention mechanism. In this seq2seq framework, the language and vision are encoded as input and an action sequence is decoded as output. Later there are many works devoted to improving the performance of VLN tasks from various aspects. And we summarize existing approaches to VLN in Figure 2.1.

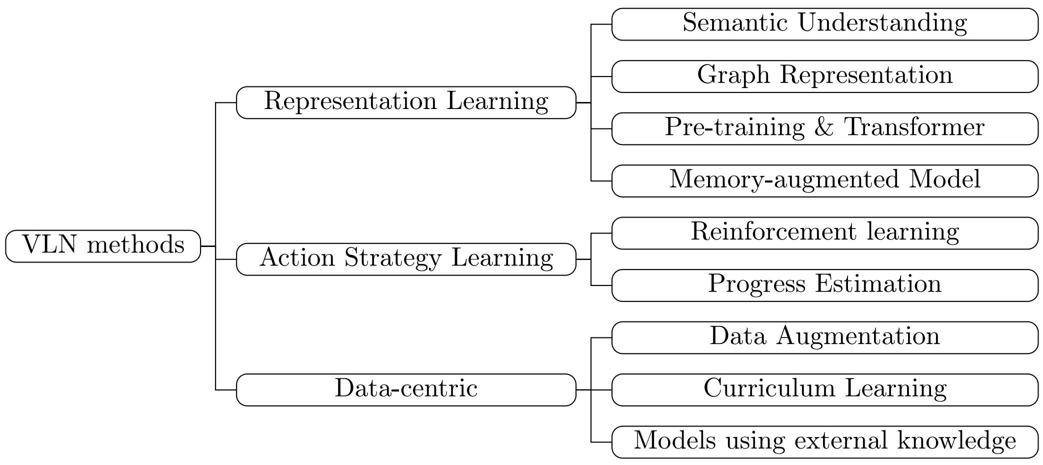


FIGURE 2.1. Categories of VLN methods.

Representation Learning

Representation learning methods aim to enable the agent to comprehend the relations between multiple modalities in VLN tasks. Effective representation learning methods aid in capturing the complex dependencies and correlations between these modalities, enabling agents to derive meaningful insights and make informed decisions.

- **Semantic Understanding** Qi *et al.* [94] introduce an object-and-action action aware model (OAAM), which contains three parts: object-aware module, action-aware module, and adaptive combination module. It proposes to decouple action and object-related sub-instructions to facilitate the learning of action-orientation alignment and visual-textual alignment. An *et al.* [2] take into account information from neighbor views to overcome the single view limit when aligning visual-textual counterparts.

- **Graph Representation** Hong *et al.* [36] further models the relationship among action, orientation, and scene via a graph leading to effective representations for better action prediction. Chen *et al.* [13] propose a Structured state-Evolution (SEvol) model, which uses the graph-based feature to represent the navigation state. It contains a Reinforced Layout clues Miner (RLM) and a Structured Evolving Module (SEM). The dual-scale graph transformer (DUET) [17] employs graph transformers to encode the topological map for long-term action planning and learn cross-modal relations with the instruction.

- **Pre-training & Transformer** Inspired by the great success of Vision-Language BERT pre-training on several visual-textual matching tasks, such as image-text retrieval [57] and referring expression grounding [129], several pre-training methods have been proposed for VLN [29, 30, 42, 80]. PRESS [61] fine-tunes the pre-trained language model BERT to obtain the textual representation for the agent. VLN-BERT [80] pre-trains its model by predicting the compatibility of a pair of instruction and visual trajectory. In the downstream tasks, it formulates the navigation as a trajectory selection problem. Lin *et al.* [68] introduce two pre-training tasks: Scene Grounding task to learn where to navigate and Object Grounding task to learn what object to localize. In addition, a memory-augmented action decoder is proposed to fuse grounded textual and visual representation. AirBERT [29] further adopts a binary classification task to predict whether the given instruction and visual trajectory are paired. These methods discard navigating action prediction during pre-training, weakening the relationship between the learned representation and the final goal: navigation action prediction. By contrast, PREVALENT [30] introduces a single-step action prediction task, aiming to learn action-oriented generic visiolinguistic representation, which can be applied to the greedy search VLN. However, PREVALENT largely overlooked the important historical context in pre-training. It only takes the static panoramic image of a single step as visual input, while failing to take into account the history trajectory information. LOViS [131] designs two specific VLN pre-training tasks: an Orientation Matching task to predict the current orientation, and a Vision Matching task to predict whether the current visual information matches with instruction. At the fine-tuning stage, it also includes an orientation module and a vision module to capture the orientation and visual information. Different conventional methods of fine-tuning, Liang *et al.* [63] introduce prompt tuning techniques to improve learning efficiency.

Recently, VLBERT-based methods significantly improve performance on VLN tasks. Hong *et al.* [39] develop a recurrent model VLN↻BERT that reuses the [CLS] token to maintain the history information. Qi *et al.* [93] propose Object-and-Room Informed Sequential BERT (ORIST), which is an object-informed sequential BERT to encode visual perceptions and linguistic instructions. It contains three components: the object-level initial embedding module, the sequential BERT module, and the room-and-direction multi-task module. Similar to VLN↻BERT, the Scene-and object-aware transformer (SOAT) includes object features as additional input to support object-level processing.

- **Memory-augmented Model** Zhu *et al.* [139] introduce the Cross-modal Memory Network (CMN) as a solution for effectively remembering and comprehending the diverse and relevant information associated with historical navigation actions. It contains three memory modules: language memory module, vision memory module, and Cross-modal Memory. VLN↻BERT employs a recurrent hidden state to encode the temporal context. ORIST [93] utilizes an extra LSTM module to maintain history visual and language information. E.T. [89] encodes the whole history of visual observations and actions into a multimodal transformer. Similarly, HAMT [15] encodes all past panoramic observations via a hierarchical vision transformer (ViT) and explicitly stores historical information. However, this leads to time-consuming training, especially for long-horizon tasks such as RxR which is with an average of 12 navigation steps.

Action Strategy Learning

Action Strategy Learning refers to a category of methods in VLN tasks that focus on learning and optimizing the action strategies of agents. These methods aim to develop effective strategies for agents to make informed decisions and take appropriate actions based on the instructions and their observations. Techniques such as Reinforcement Learning are commonly used to train agents to interact with the environment and adjust their actions to maximize long-term rewards. Progress estimation may also be employed to estimate the progress of agents and guide their actions during navigation. The goal of Action Strategy Learning methods is to enhance the decision-making and action-taking abilities of agents, ultimately improving their performance in VLN tasks.

- **Reinforcement learning** Reinforcement learning [64, 86] is also an important research direction to improve the navigation ability of agents. Wang *et al.* [123] proposed a Reinforced Planning Ahead (RPA) model, which is the first to combine model-free and model-based deep reinforcement learning (DRL) for VLN tasks. Later, Wang *et al.* [122] introduce a Reinforced Cross-modal Matching method that utilizes reinforcement learning and imitation learning methods. They design a reasoning navigator to learn the cross-modal grounding and also propose a cycle-reconstruction reward to let the agent better understand the language input. Tan *et al.* [116] also train the agent by mixing imitation learning (IL) and reinforcement learning (RL) to combine the benefits from off-policy and on-policy optimization. Further, a Soft Expert Reward Learning (SERL) model *et al.* [120] is proposed to learn adaptive rewards instead of hand-crafted hard rewards. The model contains two parts: a Soft Expert Distillation (SED) module and a Self Perceiving (SP) module. The SED module aims at encouraging the agent to behave as an expert and the SP module lets the agent towards the target location as fast as possible.

- **Progress Estimation** Ke *et al.* [49] present the Frontier Aware Search with backTracking (FAST) Navigator, which utilizes asynchronous search to let the agent could backtrack when it is detected as lost. This method aims at alleviating the expense of beam search. Without beam search, Ma *et al.* [76] introduce a self-monitoring agent, which uses greedy decoding selection with one condition. It contains two complementary modules: a visual-textual co-grounding module and a progress monitor. The visual-textual co-grounding module performs grounding across visual and textual modalities, and the progress monitor reflects the progress towards the goal via estimating the completeness of instruction-following. When the progress monitor output decreases, the agent will move back to the last viewpoint and select the action. Similarly, Ma *et al.*

[77] propose a regretful navigation agent. It contains a regret module to allow the agent to learn when to roll back to the previous location, and a progress marker to avoid going to a visited location. Zhu *et al.* [136] propose the Auxiliary Reasoning Navigation (AuxRN) framework, which contains four auxiliary reasoning tasks: a trajectory retelling task to explain previous actions, a progress estimation task, and an angle prediction task, and a cross-modal matching task. These tasks cooperate to explore the semantic meaning of visual features and enhance the reasoning ability of agents.

Data-centric

Data-centric methods focus on utilizing data to enhance the agents' performance. These methods involve techniques such as data augmentation, curriculum learning, and leveraging additional knowledge sources. Data augmentation techniques aim to increase the diversity and quantity of training data to improve model robustness and generalization. Curriculum learning gradually increases the difficulty of training tasks to enhance learning effectiveness. Leveraging additional knowledge involves incorporating external information to enrich training data and enhance model performance.

- **Data Augmentation** To overcome the limited seen environments and enhance the generalisability to unseen environments, several works concentrate on augmenting the training dataset, such as instruction-trajectory pairs and environments. For instruction-trajectory pairs, Speaker-Follower [24] model is introduced to address the data scarcity problem and augments the instruction-trajectory pairs. It synthesizes new instructions for randomly sampled trajectories. It designs a language model as a speaker to learn the relationship between vision and language information thus could synthesis new instructions, and a follower network to take action. Kamath *et al.* [48] constuct a large-scale dataset with 4.2M instruction-trajectory pairs, which is larger than existing human-annotated datasets. To build the dataset, they use a high-quality multilingual navigation instruction generator Marky [121] to generate instructions, and utilize an image-to-image GAN [50] to synthesize image observations. For environments, Tan *et al.* [116] propose EnvDrop, which is an environmental dropout strategy to generate new environments (*i.e.* paths and instructions). The Random Environmental Mixup (REM) method [69] leverages mixup environments to generate augmented data with cross-connected house scenes. For each scene, REM identifies key viewpoints based on the room connection graph. Then it constructs augmented scenes by cross-connect the key views from different scenes and it generates augmented instruction path pairs. ENVEDIT [59] creates new environments by editing the training environment in three diverse aspects: style, object appearance, and object classes.

- **Curriculum Learning** Some works concentrate on transferring the agent's navigation ability from shorter instructions to longer instructions and utilize the curriculum-based training paradigm on VLN tasks. Basically, curriculum learning is a concept of beginning with simpler aspects of a task and progressively escalating the difficulty level. Wang *et al.* [138] propose BabyWalk to compose long instructions into shorter ones, which is the first to apply Curriculum Learning in VLN tasks. The learning process comprises two phases: imitation learning to achieve BabySteps and curriculum-based reinforcement learning for maximizing rewards on increasingly complex navigation tasks with longer instructions. Similarly, Hong *et al.* [37] build the Fine-Grained Room-to-Room dataset (FGR2R), which segments the long instructions into sub-instructions. They employ a heuristic method based on grammatical relations provided by the Standford NPL Parser [92] to segment lengthy and complex instructions into shorter sub-instructions. Then a shifting module determines the completion status of each sub-instruction, ensuring that only one sub-instruction is accessible to the agent at each time step for textual grounding. Different directly separate the long instructions into short instructions, Zhang *et al.* [130] introduce the self-paced curriculum learning (SPCL) to incorporate human prior knowledge and re-arrange the benchmark Room-to-Room datasets to make it suitable for curriculum learning.

- **Models using external knowledge** There are several works employing external knowledge to assist navigation. To incorporate commonsense knowledge for navigation, the Cross-modality Knowledge Reasoning (CKR) model [25] proposes a Knowledge-enabled Entity Relationship Reasoning (KERR) module that incorporates the external knowledge from ConceptNet [110] to assist room entity and object entity reasoning. ADAPT [65] first uses the Contrastive Language-Image Pre-training (CLIP) model to build an Action Prompt Base. During the navigation process, the agent retrieves the relevant action prompts and then concatenated with instruction encoding to obtain the prompt-based instruction features.

Chapter 3

History-Enhanced and Order-Aware Pre-training

In order to address the challenge posed by the lack of a pre-trained model specifically designed to emphasize temporal information for the VLN task, In this Chapter, we propose a history-enhanced and order-aware pre-training with the complementing fine-tuning paradigm for the VLN task. This paradigm aims to enhance the temporal understanding capabilities of models utilized in VLN by incorporating historical and temporal information.

3.1 Introduction

Vision-and-Language Navigation (VLN) has attracted large attention in recent years. It lies in the interaction of computer vision, natural language processing, and robotics and has great importance in real-world applications. VLN task requires an agent to follow a natural language instruction and navigate in 3D simulated environments rendered by realistic images. Generally speaking, VLN tasks could be categorized into three types: (I) VLN with detailed navigation instructions such as R2R [5] and RxR [54]; (II) VLN with high-level, concise instructions for remote object grounding, such as REVERIE [95] and SOON [135]; (III) VLN with communication dialogs, such as NDH [117].

Inspired by the great success of pre-trained Vision-and-Language Transformer-based models, which demonstrate the great effectiveness of modeling cross-modality correspondence. Recently, there are many pre-training strategies for VLN tasks [80, 30, 29, 15]. VLN-BERT [80] pre-trains its model by predicting the compatibility of a pair of instructions and visual trajectory. It selects the best matching trajectory from several candidate paths. AirBERT [29] further adopts a binary classification task to predict whether the given instruction and visual trajectory are paired. Both VLN-BERT and AirBERT discard navigating action prediction during pre-training, weakening the relationship between the learned representation and the final goal: navigation action prediction. By contrast, PREVALENT [30] introduces a single-step

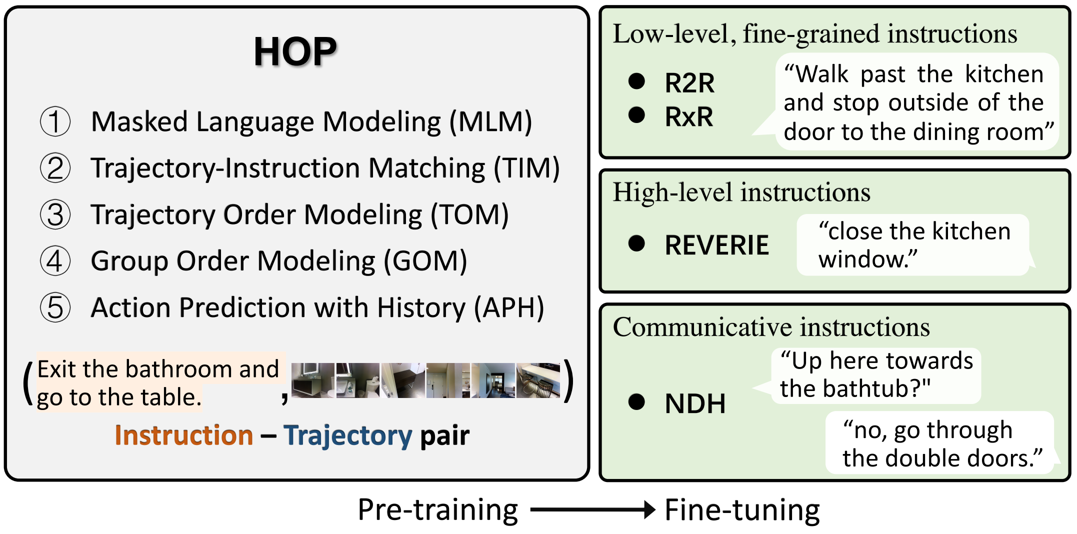

HOP
① Masked Language Modeling (MLM)
② Trajectory-Instruction Matching (TIM)
③ Trajectory Order Modeling (TOM)
④ Group Order Modeling (GOM)
⑤ Action Prediction with History (APH)
(Exit the bathroom and go to the table.,
Instruction – Trajectory pair
Low-level, fine-grained instructions
R2R
RxR
"Walk past the kitchen and stop outside of the door to the dining room"
High-level instructions
REVERIE
"close the kitchen window."
Communicative instructions
NDH
"Up here towards the bathtub?"
"no, go through the double doors."
Pre-training
Fine-tuning

FIGURE 3.1. Illustration of the proposed pre-training and fine-tuning paradigm for VLN. The model is pre-trained with five proxy tasks, and fine-tuned on four downstream VLN tasks: R2R, RxR, REVERIE and NDH (detailed in Section 5.3).

action prediction task, aiming to learn action-oriented generic visiolinguistic representation, which can be applied to the greedy search VLN. However, PREVALENT largely overlooked the important historical context in pre-training. It only takes the static panoramic image of a single step as visual input, while failing to take into account the history trajectory information. Indeed, VLN is a Partially Observable Markov Decision Process (POMDP), where the agents rely heavily on past experiences for making future action decisions. Furthermore, VLN is a spatiotemporal task that is sensitive to the sequence order of the trajectory. Thus the ability of temporal order reasoning is also beneficial to action decision-making. Nevertheless, all the above three methods do not mine temporal order information from instructions or visual observations explicitly.

To address the above-mentioned issues in pre-training, we propose a novel history-and-order aware pre-training paradigm (HOP) to enhance the learning of visual-textual correspondence for VLN tasks. **First**, we provide history visual observations to the action prediction task, called Action Prediction with History (APH), which helps the model locate the sub-instruction to be executed and thus improve the action prediction accuracy. **Second**, we design two order-aware proxy tasks, Trajectory Order Modeling (TOM) and Group Order Modeling (GOM). Given an instruction, TOM requires the model to recover the order of shuffled visual trajectory from a fine-grained level, and GOM requires the model to predict the order of two groups of sub-trajectories from a coarse level. These two tasks explicitly equip the model with the ability to understand the temporal order within instructions, in addition to the visual-textual matching capability. The overall proposed pre-training and fine-tuning tasks are illustrated in Figure 3.1.

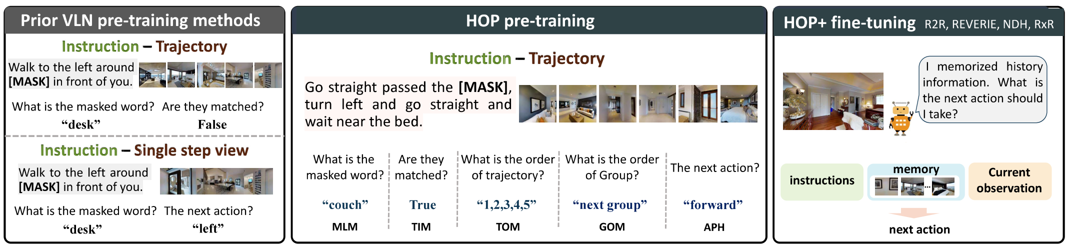

Prior VLN pre-training methods
Instruction – Trajectory
Walk to the left around [MASK] in front of you.
What is the masked word? Are they matched?
"desk" False
Instruction – Single step view
Walk to the left around [MASK] in front of you.
What is the masked word? The next action?
"desk" "left"
HOP pre-training
Instruction – Trajectory
Go straight passed the [MASK], turn left and go straight and wait near the bed.
What is the masked word? "couch" MLM
Are they matched? True TIM
What is the order of trajectory? "1,2,3,4,5" TOM
What is the order of Group? "next group" GOM
The next action? "forward" APH
HOP+ fine-tuning R2R, REVERIE, NDH, RxR
I memorized history information. What is the next action should I take?
instructions
memory
Current observation
next action

FIGURE 3.2. Overview of our pre-training and fine-tuning paradigm for VLN. Our model is pre-trained with five proxy tasks: Mask Language Modeling (MLM), Trajectory-Instruction Matching (TIM), Trajectory Ordering Modeling (TOM), Group Ordering Modeling (GOM), and Action Prediction with History (APH). We also devise an external memory network for fine-tuning to better utilize history context..

Although HOP has largely boosted the navigation performance on several downstream tasks, we find its potential is limited by the processing inconsistency of history information between the pre-training and fine-tuning stages. During pre-training, our preliminary work HOP takes all the past visual observations as history input tokens for the transformer. But during the fine-tuning stage, HOP utilizes only a single state vector to hold history information that is updated at each navigation step, thus making it easy to forget long-term memories. A naive solution is to use the same history input during fine-tuning. Although this is demonstrated effective in our later experiments, the improvement is slight and it causes huge undesired computational costs (the longer the trajectory, the more computational burden). In addition, the history trajectory may contain misleading information. Specifically, if an agent takes a wrong step, then this historical trajectory is not helpful for decision-making. Thus, how to dynamically select effective history information remains a challenging task.

To address this issue, in this work, we extend our HOP to HOP+ by designing a memory network for fine-tuning, which effectively selects and summarizes useful historical information adaptively for action prediction. The Overview of our pre-training and fine-tuning paradigm for VLN are as shown in Figure 3.2 Thus, the agent is able to access all the memories at each step and almost does not increase large computation to the transformer. Specifically, the memory network consists of an attention module and a memory update module. The network first uses an attention-based GRU to learn relevant information from historical trajectories based on current observations and instruction, and then stores them in memory and updates them at each time step. The output feature of the memory network and the output features of the vision encoder are concatenated as the input of the cross-modal encoder to make final decisions.

Finally, we conduct extensive experiments on various VLN tasks, including R2R [5], REVERIE [95], RxR [54], and NDH [117]. These tasks are characterized by different aspects, so the validity of the model can be verified from multiple aspects. R2R is an in-domain task to verify the performance of the agent in an unseen environment. The other tasks are out-of-domain tasks, which aim at validating the generalization performance of the agent for new tasks. RxR has much longer instructions and is challenging to understand for the agent. NDH requires the agent to reach the destination based on the historical dialogue. REVERIE gives concise and high-level instructions, focusing on grounding target objects. Extensive experiments on these downstream tasks illustrate the superiority of our HOP+ over other methods and achieve state-of-the-art performance. In addition, our ablation study shows that the proposed memory network can save computational costs and achieve better performance. Our final model obtains 60% in SPL on R2R, 3.90 in GP on NDH, 28.24% in SPL (16.86% in RGSPL) on REVERIE, and 0.36 in sDTW on RxR.

3.2 Background

3.2.1 Pre-training

By pre-training on large amounts of diverse unlabeled data, models can learn representations that capture general semantic and contextual information. These learned representations can then be fine-tuned on smaller labeled datasets or on specific tasks, where the model is adapted to task-specific data. Self-supervised learning aims to create artificial supervision signals from unlabeled data itself, effectively transforming unsupervised learning problems into supervised learning problems. Transformer-based architectures, such as BERT (Bidirectional Encoder Representations from Transformers) [22], has gained significant attention due to their ability to capture long-range dependencies and contextual relations in sequential data. The self-attention mechanism lets the model focus on relevant information in the input sequence, resulting in a more comprehensive and context-aware representation. The effectiveness of pre-training has been demonstrated on a wide range of computer vision and natural language processing tasks [62, 18, 115], such as image-text retrieval, Image Captioning, and Visual Question Answering (VQA).

3.2.2 Vision-and-Language Navigation Pre-training

With the success of large-scale pre-training for vision-and-language [74, 115, 18, 62], a variety of pre-training methods [80, 15, 61] have been proposed to handle VLN tasks and achieve state-of-the-art performance. Majumdar *et al.* [80] propose VLN-BERT, which learns the matching degree between the instruction and panoramic image sequence. Guhur *et al.* [29] builds a large-scale in-domain pre-training dataset BnB from online rental marketplaces. However, these pre-training tasks remain similar to general vision-and-language tasks, such as image-text matching, while ignoring the important action decision in the VLN task. Then Hao *et al.* [30] introduces the task of Action Prediction (AP) in PREVALENT. While it still ignores the historical

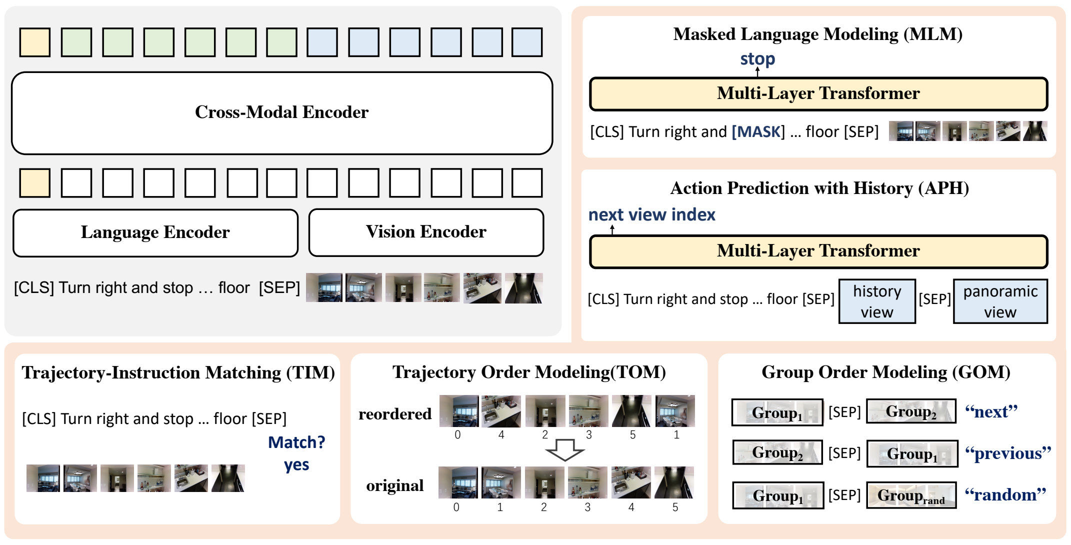
TABLE 3.1. Comparison with previous VLN pre-training works.

	HOP+ (proposed)	VLN-BERT [80]	PREVALENT [30]	Airbert [29]
Dataset	Augmented R2R dataset Processed BnB dataset	Conceptual Captions [105] Wikipedia and BookCorpus R2R dataset	Augmented R2R dataset	Conceptual Captions [105] BnB dataset
Visual Input	Trajectory	Trajectory	Panoramic view (single step)	Trajectory
Objectives	Masked Language Modeling Action Prediction with History Trajectory-Instruction Matching Trajectory Order Modeling Group Order Modeling	Masked Language Modeling Image-Caption matching Trajectory-Instruction matching	Masked Language Modeling Action Prediction	Masked Language Modeling Image-Caption matching Trajectory-Instruction matching (shuffling loss)
Pre-training Device	4 V100 GPUs	-	8 V100 GPUs	8 V100 GPUs
Downstream task	R2R, REVERIE, RxR, NDH	R2R	R2R, NDH, HANNA	R2R, REVERIE

trajectory and temporal information. In addition, VLN-BERT and PREVALENT do not model temporal order contained in both instructions and trajectories, and thus are weak at temporal reasoning. To alleviate this problem, ALTR [42] propose a “Next Visual Scene” task to predict the visual features of future steps, which only connects two consecutive steps. AirBERT tries to distinguish aligned instruction and trajectory from shuffled ones at a coarse level, which does not take full advantage of the temporal correspondence in training data. Our preliminary work HOP [96] enhances the learning of temporal order modeling and historical information by introducing three VLN-specific proxy tasks for pre-training. These tasks make the model understand the historical context and temporal order information while learning visual-textual correspondence, which facilitates the final action prediction. We compare our method HOP+ with previous VLN pre-training methods, as shown in table 3.1,

3.2.3 Memory Networks

Memory networks have been widely used in a variety of research areas, such as Question Answering [111, 55], Visual Question Answering (VQA) [126, 75], and Visual-and-Language Navigation [139]. The earliest work can be traced back to Neural Turing Machine [27] and Memory Neural Network [46]. They both propose an external memory with a read-write mechanism. Sukhbaatar *et al.* [111] propose an end-to-end Memory Network (MemNN) for the Question and Answering task, which first stores all the sentence inputs as facts and then retrieves relevant memory blocks based on the given question to output answer. Ma *et al.* [75] apply Memory-Augmented Networks to VQA tasks to learn uncommon question-answer pairs. It uses visual and question features as augmented memories and utilizes an LSTM controller to determine when to write and read the memory, thus selectively paying more attention to scarce training items. Recently, Zhu *et al.* [139] introduce a Cross-modal Memory Network to specifically address the task of Navigation from Dialog History. It concentrates on memorizing the dialog history and panoramic views while neglecting the historical trajectory information. Our proposed HOP+ introduces an external memory network at the fine-tuning stage to exploit historical information from pre-training in order



Cross-Modal Encoder
Language Encoder
Vision Encoder
[CLS] Turn right and stop … floor [SEP]
Masked Language Modeling (MLM)
stop
Multi-Layer Transformer
[CLS] Turn right and [MASK] … floor [SEP]
Action Prediction with History (APH)
next view index
Multi-Layer Transformer
[CLS] Turn right and stop … floor [SEP]
history view
[SEP]
panoramic view
Trajectory-Instruction Matching (TIM)
[CLS] Turn right and stop … floor [SEP]
Match?
yes
Trajectory Order Modeling(TOM)
reordered
0 4 2 3 5 1
original
0 1 2 3 4 5
Group Order Modeling (GOM)
Group1 [SEP] Group2 "next"
Group2 [SEP] Group1 "previous"
Group1 [SEP] Grouprand "random"

FIGURE 3.3. The main architecture of our pre-training model and five proxy tasks.

to assist the agent in action prediction. To be specific, our HOP+ consists of an attention-based module and a memory update module. Our proposed memory model differs from the aforementioned memory networks in several ways. Firstly, it uses historical observations as facts and the fact changes dynamically as the agent navigates, rather than as a static input. Secondly, we utilize a memory update module to update the memory during navigation. Furthermore, our HOP+ can be applied to different downstream tasks instead of one specific task.

3.3 Method

3.3.1 Pretrain Model Architecture

The model architecture is illustrated on the top-left of Figure 3.3, which is similar to LXMERT [115]. Taking the instruction-trajectory pair as input, the model first utilizes a language encoder and a vision encoder to extract single-modal representations from the instruction and image sequence, respectively. Then, these representations are fed into a cross-modal encoder to implement interactions between the two modalities and generate the final fused representations.

Language Encoder

We first use WordPieces [47] to tokenize all words in an instruction, obtaining a sequence of tokens: $\texttt{[CLS]}, w_1, w_2, \ldots, w_L, \texttt{[SEP]}$, where [CLS] and [SEP] are added special tokens. Then, the text embedding of each token is obtained via summing up the token embedding and the position embedding, followed by Layer Normalization (LN). At last, the text embedding is passed through the single-modal language encoder, of which each layer consists of a self-attention sub-layer and a feed-forward sub-layer. The outputs of the language encoder are used as language features.

Vision Encoder

Trajectory $\tau = \langle v_1, v_2, \dots v_T \rangle$ represents the image sequences observed by the agent when traversing the environment, where v_i is the observed image of the environment at step i and T is the number of total steps. To better capture order information from the trajectory, we use the front view image of the agent's observation at each position, rather than using the panoramic image. This is because panoramic images of the adjacent observation points in the same room are similar, causing difficulties for the agent to explore the dynamic and temporal information of the entire trajectory.

We first use ResNet-152 [33] pre-trained on ImageNet [103] to extract a 2048-dimensional image feature vector v_{vis} for each front view image v_i. Then, we compute the orientation feature of heading α and elevation β as $[\sin\alpha; \cos\alpha; \sin\beta; \cos\beta]$, and repeat it for 32 times to constitute a 128-dimensional direction feature vector v_d as same as [116]. Each image v_i in the trajectory is finally represented by a 2176-dimensional feature vector $v_i = [v_{vis}; v_d]$ by concatenating v_{vis} and v_d. At last, the image features of trajectory τ are passed through the single-modal vision encoder, of which each layer consists of a self-attention sub-layer and a feed-forward sub-layer. The outputs of the vision encoder are used as vision features.

Cross-Modal Encoder

We use the Cross-Modal Encoder to fuse features from both language and vision modalities. For the cross-modal encoder, each layer contains two self-attention sub-layers, one bi-directional cross-attention sub-layer and two feed-forward sub-layers. The outputs of the cross-modal encoder are used as cross-modal features for pre-training and downstream tasks.

Following [30], we set the layers' number N_{text}, N_{image}, N_{cross} of text encoder, vision encoder, and cross-modal encoder to 9, 1, and 3, respectively.

3.3.2 Pre-training Proxy Tasks

Masked Language Modeling (MLM)

As shown in Figure 3.4, the input instruction tokens $w = \langle w_1, w_2, \dots, w_L \rangle$ is randomly replaced by a special token `[mask]` with probability 15%. The output feature is trained to predict masked words w_m based on surrounding words $w_{\backslash m}$ and the image trajectory $\tau = \langle v_1, v_2, \dots, v_T \rangle$, by minimizing the negative log-likelihood:

$$\mathcal{L}_{\text{MLM}}(\theta) = -\mathbb{E}_{(w,\tau)\sim D} \log P_\theta(w_m | w_{\backslash m}, \tau)\,, \tag{3.1}$$

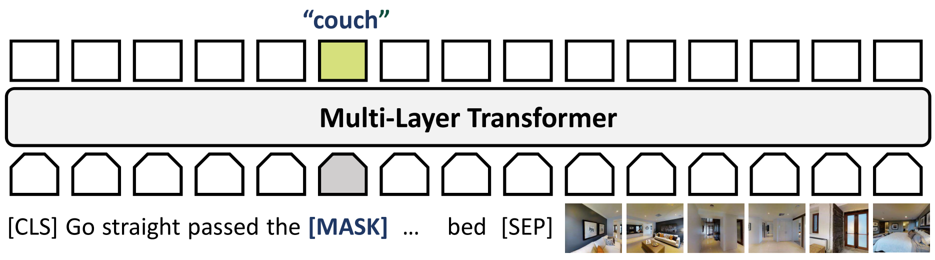


FIGURE 3.4. Illustration of the Masked Language Modeling (MLM) task.

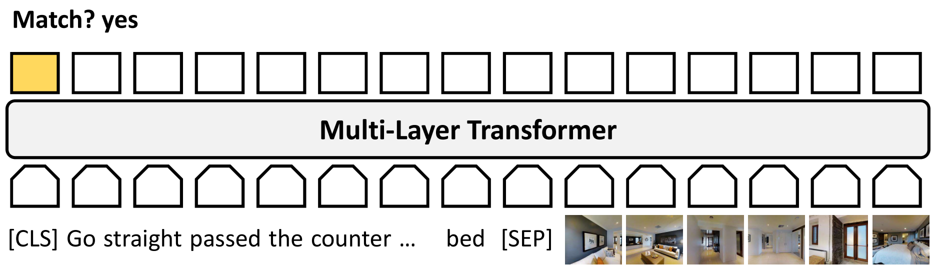


FIGURE 3.5. Illustration of the Trajectory-Instruction Matching (TIM) task.

where θ represents trainable parameters. And each pair (w, τ) is sampled from the training set D.

Trajectory-Instruction Matching (TIM)

TIM task aims at training a binary classifier to predict whether the image trajectory and instruction are matched. As illustrated in Figure 3.5, the output on the special token `[CLS]` is the fused instruction-pairs (w, τ) representation. We use a fully connected (FC) layer to calculate the matching score $s_\theta(w, \tau)$. We apply the binary cross-entropy loss for optimization:

$$\mathcal{L}_{\text{TIM}}(\theta) = -\mathbb{E}_{(w,\tau)\sim D}[y \log P_\theta + (1-y) \log P_\theta], \tag{3.2}$$

where $P_\theta = s_\theta(w, \tau)$, and $y \in \{0, 1\}$ is the binary label that indicates if the sampled pair is a match.

In addition, to make the model focus on the differentiation between paths, we randomly generate negative samples (*i.e.* , mismatched trajectory) in the same environment, with a probability of 50%.

Trajectory Order Modeling (TOM)

A good VLN agent should not only understand visual-textual correspondence but also be able to be aware of the sequence order among the correspondence. The motivation for the TOM task is to make the model reconstruct the correct order of the visual trajectories based on given instructions, thus mining temporal order information. Here

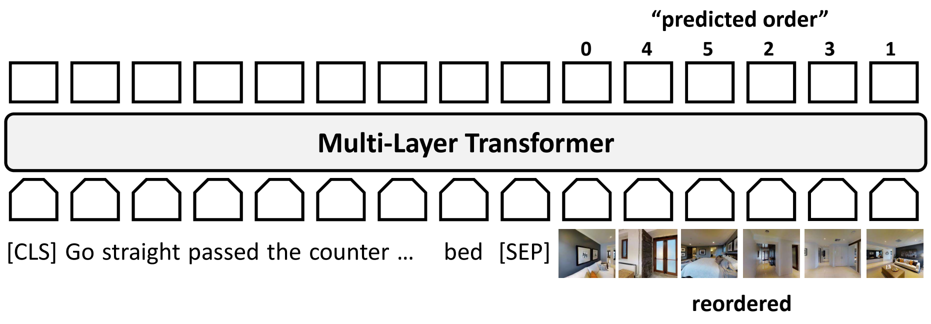


FIGURE 3.6. Illustration of the Group Order Modeling (TOM) task.

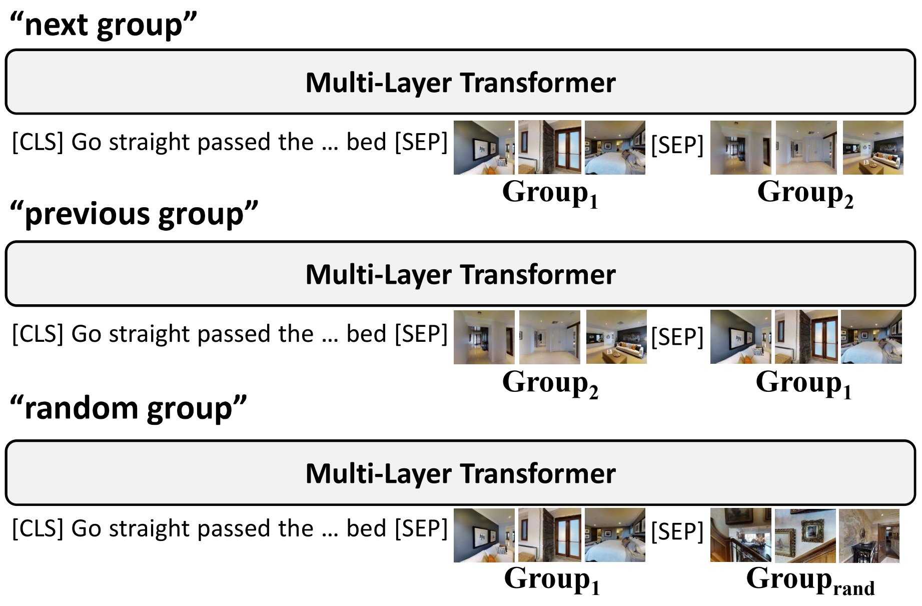


FIGURE 3.7. Illustration of the Group Order Modeling (GOM) task.

we randomly shuffle 50% images of the trajectory. As is shown in Figure 3.6, the inputs are instruction w and reordered trajectory τ'. The output vision feature of the cross-modality encoder is fed into an FC layer that predicts the order r'_k of each image k. This task is optimized by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{TOM}}(\theta) = -\mathbb{E}_{(w,\tau')\sim D} \sum_{i=1}^{N} y_i \log P_\theta(r'_k|w, \tau')\,, \tag{3.3}$$

where N is the number of steps of the trajectory, $y_i = 1$ indicates the predicted order r'_k for image k is the original order i, and otherwise $y_i = 0$.

Group Order Modeling (GOM)

The goal of the GOM task is to make the model learns temporal information from the level of sub-trajectories. As shown in Figure 3.7, the inputs are the instruction w and

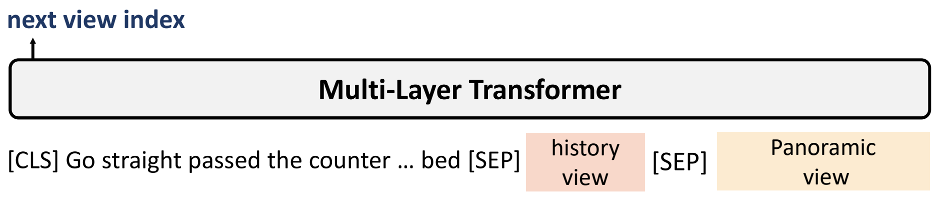


FIGURE 3.8. Illustration of the Group Order Modeling (APH) task.

image sequence group (G_1, G_2). The trajectory τ is divided into two sequence groups in an even way. We randomly choose to keep G_2 after G_1, place before G_1, or replace it with image sequence groups sampled from other trajectories, all with a probability of $1/3$. Thus GOM is formulated as a triple classification problem, which predicts the previous, next, or random relation between two sub-trajectories. $c = 1$ denotes that G_1 is before G_2; $c = 2$ indicates that G_1 comes after G_2; $c = 3$ indicates that G_2 is a random set of image sequences from other trajectories. We use the special token `[SEP]` to distinguish between the two groups of trajectories. The output on the `[CLS]` indicates the fused representation of the input visual and textual information, and then be fed into an FC layer with softmax function to predict c'. We optimize the GOM task via the cross-entropy loss:

$$\mathcal{L}_{\text{GOM}}(\theta) = -\mathbb{E}_{(w,(G1,G2))\sim D} \sum_{c} y_c \log P_\theta(c'|w, (G1, G2)), \tag{3.4}$$

where $y_c \in \{0, 1\}$ denotes whether the predicted class c' is the desired class c or not.

Action Prediction with History (APH)

As shown in Figure 3.8, The inputs are the instruction w, the history trajectory $\tau_{t-1} = \langle v_1, v_2, \ldots, v_{t-1} \rangle$, and the panoramic view $\boldsymbol{p} = \{p^1, p^2, \ldots, p^{36}\}$ of the current step t. We concatenate the history trajectory τ_{t-1} and panoramic view $\boldsymbol{p}$ as vision input. APH is considered a classification problem and the action prediction is performed by selecting the next view image v'_{t+1} from the candidate views. We feed the fused representation of `[CLS]` into an FC layer to predict the next view view v'_{t+1}, by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{APH}}(\theta) = -\mathbb{E}_{(w,\tau,v_{pano})\sim D} \sum_{p} y_p \log P_\theta(v'_{t+1}|w, \tau_{t-1}, v_{t_p}), \tag{3.5}$$

where p indicates labels of the 36 views in the panoramic view image, and $y_p \in \{0, 1\}$ denotes whether the predicted next view image v'_{t+1} is the desired one.

3.3.3 Pre-training Datasets

We construct our pre-training dataset based on existing datasets: PREVALENT [30] and BnB [29]. PREVALENT uses a pre-trained speaker model to produce more instructions to augment R2R dataset. It contains 104K original R2R samples and 6482K synthesized samples. BnB dataset collects image-caption pairs from Airbnb.

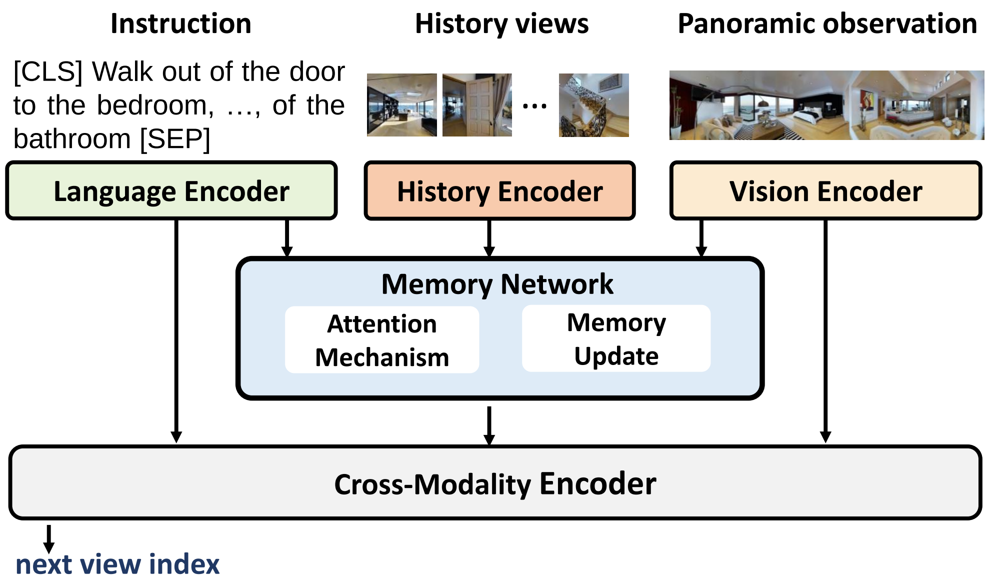


FIGURE 3.9. Overview of fine-tuning with memory network.

We use raw images and captions from the BnB dataset and reprocessed them. Indeed, nearly half of the BnB images are captionless (*i.e.* images without captions). Thus, to better adapt BnB dataset to the designed pre-training tasks such as Trajectory Order Modeling, we remove these captionless images. To construct path-instruction pairs, we concatenate the images and concatenate the corresponding captions. Each path contains 5-7 images, which is consistent with the R2R dataset. For image features, we used a Resnet-152 network pre-trained on ImageNet to extract a mean-pooled feature vector, the same as the encoding method of images in Matterport3D. Our processed BnB data contains 342K image sequence-caption pairs.

3.3.4 Fine-tuning with Memory Network

During fine-tuning, our preliminary work HOP [96] uses the state vector as historical information and updates it at each step. This method is more dependent on the state of the previous step and thus easily forgets long-term memories. While if all the observations of previous steps are used as history input during fine-tuning, it will largely increase the computational cost as the trajectory becomes longer. To address this problem, we design a memory network for fine-tuning (Figure 3.9), which selects and summarizes historical information for action prediction during fine-tuning.

The overview of finetuning with the memory network is illustrated in Figure 3.9, where the architecture of the language encoder, vision encoder, and cross-modality encoder is the same as that of the pre-training stage. We additionally introduce a history encoder that shares the same structure with a vision encoder to encode historical trajectory information, and a memory network to store and update historical information for action prediction.

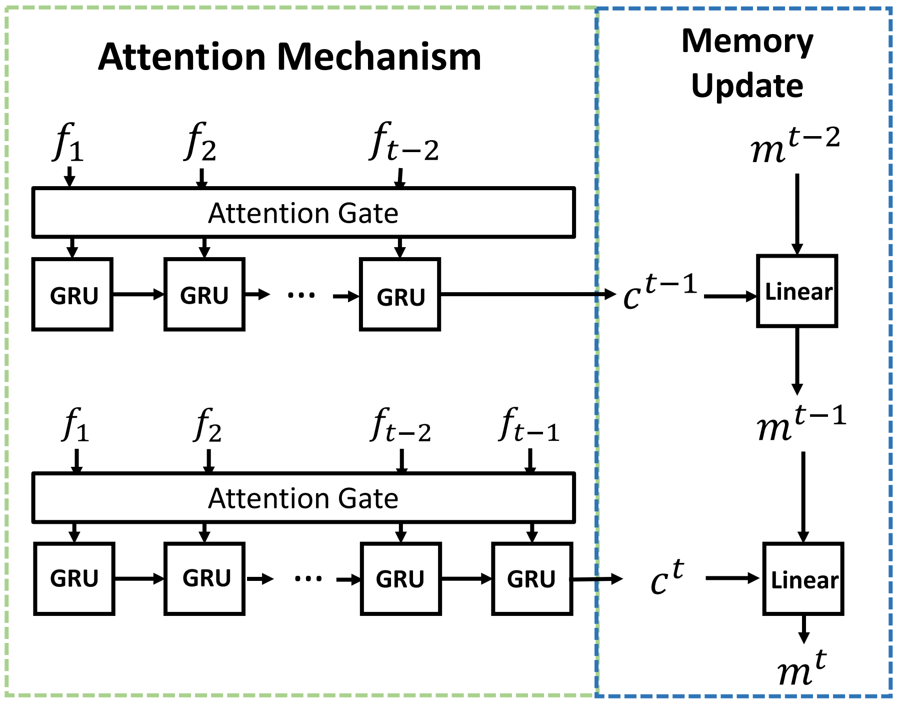


FIGURE 3.10. Details of the memory network.

To be specific, the inputs are the instruction w, the history views $\tau_{t-1} = \langle v_1, v_2, \ldots, v_{t-1} \rangle$, and the panoramic observation $\boldsymbol{p}_t = \{p_t^1, p_t^2, \ldots, p_t^{36}\}$ of current time step t. The output features of the instruction from the language encoder are pooled as global representation Q, and the output features of panoramic view images $\boldsymbol{p}_t = \{p_t^1, p_t^2, \ldots, p_t^{36}\}$ from vision encoder are pooled as global representation P_t. Then, W and P_t are concatenated as a query q for the memory network. Meanwhile, we use the history encoder to encode historical trajectory before the current time step t, which shares the same architecture but with different parameters as the vision encoder. This requires fewer computation resources than directly passing the concatenated trajectory and panoramic observation features into a single vision encoder to implement self-attention. The output features of the history encoder are then passed into the memory network as input facts $F = \{f_1, ..., f_{t-1}\}$ for memory updating.

Memory Network

As shown in Figure 3.10, the memory network is comprised of an attention-based GRU to implement an internal attention mechanism and a linear layer to implement a memory update mechanism. To produce a contextual vector c^t for memory updating, the attention-based GRU computes a scalar attention gate g_i^t for each fact f_i according to the query q and the previous memory m^{t-1}:

$$\begin{aligned} z_i^t &= [f_i \circ q; f_i \circ m^{t-1}; |f_i - q|; |f_i - m^{t-1}|] && (3.6) \\ Z_i^t &= W^{(2)} \tanh\left(W^{(1)} z_i^t + b^{(1)}\right) + b^{(2)} && (3.7) \\ g_i^t &= \frac{\exp(Z_i^t)}{\sum_{k=1}^{t-1} \exp(Z_k^t)}. && (3.8) \end{aligned}$$

where $\circ$ indicates the elementwise product, $[;]$ denotes concatenation on feature channel, and $|\cdot|$ denotes the elementwise absolute function.

Then, the attention gate g_i^t is applied in the GRU to generate contextual vector c^t:

$$\begin{aligned} r_i &= \sigma\left(W^{(r)} f_i + U^{(r)} h_{i-1} + b^{(r)}\right) && (3.9) \\ \tilde{h}_i &= \tanh\left(W^{(h)} f_i + r_i \circ U^{(h)} h_{i-1} + b^{(h)}\right) && (3.10) \\ h_i &= g_i^t \circ \tilde{h}_i + (1 - g_i^t) \circ h_{i-1} && (3.11) \\ c^t &= h_{t-1} && (3.12) \end{aligned}$$

where W, U and b are learnable parameters, h is the hidden state of the GRU network. The contextual vector c^t is the last hidden state of the GRU network. We update the memory by a linear layer with reference to previous memory m^{t-1}, current contextual vector c^t, and query q:

$$m^t = ReLU\left(W[m^{t-1}; c^t; q] + b\right) \tag{3.13}$$

At last, we concatenate memory token m^t and current panoramic observation tokens $\boldsymbol{p}_t$ as vision modality representation and feed it into cross-modality encoder to make decisions. In fact, the input token length of cross-modality encoder will be increased by only one due to the scalar memory m^t. Undoubtedly, this will further reduce the computation consumption compared to directly passing the concatenated trajectory and observation features from vision encoder to cross-modality encoder, which has much more tokens than combining scalar memory m^t.

3.4 Experiments

3.4.1 Experimental Setup

Datasets

We evaluate our method on four VLN tasks: VLN with low-level, fine-grained instructions (R2R [5], RxR [54]); Navigation from Dialog History (NDH) [117], and VLN with high-level instructions (REVERIE [95]).

Room-to-Room (R2R) gives detailed instructions to let the agent navigate in the indoor environment. These instructions contain rich linguistic information, such as "Exit the room using the door on the left. Turn slightly left and go past the round table an chairs. Wait there". The dataset consists of 21,567 instructions, 10,800 panoramic views, and 7,189 trajectories. The average length of instructions is 29 words.
Room-Across-Room (RxR) is an updated version of R2R. It has longer and more detailed instructions, such as "You are currently facing towards a bathtub which is in the centre of a room, turn to your left, take a few steps ...... that is your end point". In addition, the instructions are in three languages (*i.e.* English, Hindi, and Telugu). The RxR dataset contains 16,522 trajectories and 126,069 instructions, with an average length of 78 words.
REVERIE requires the agent not only to reach the goal destination but also to find a target object. The instructions are short and high-level, such as "Clean coffee table on a shag blue rug on first floor". It contains 4,140 target objects and 21,702 instructions, with an average length of 18 words.
Navigation from Dialog History (NDH) requires an agent to reach the target location based on the dialog history. Because the dialogue is unclear and ambiguous, it can be difficult for the agent to reach a destination based on the information in the dialogue. It has three settings: (1) Oracle, which uses the shortest path as ground truth observed by the Oracle; (2) Navigator, which utilizes the path adopted by the human navigator as ground truth; (3) Mixed, which takes the shortest path or the path of human if the human visits the target location. The CVDN dataset is used for the NDH task, which consists of 2,050 dialogs.

Implementation Details

Following [30], we set the layers' number N_{text}, N_{image}, N_{cross} of language encoder, vision encoder, and cross-modal encoder to 9, 1, and 3, respectively. For pre-training, the whole model is trained for 15 epochs on 4 Tesla V100 GPUs using a learning rate of 5×10^{-5} and batch size of 512. The optimizer is AdamW [73]. We sample proxy tasks for each mini-batch to train the HOP+ model. When fine-tuning on the R2R task, we use the same augmented data as [39]. The model is trained for 300,000 iterations with a batch size of 16 and a learning rate of 1×10^{-5}. When fine-tuning on NDH task, we train the model for 200,000 iterations with a batch size of 4 and a learning rate of 1×10^{-5}. When fine-tuning on REVERIE task, we train the model for 200,000 iterations with a batch size of 8 and a learning rate of 1×10^{-5}. These tasks are fine-tuned on a single 1080Ti GPU. When fine-tuning on the RxR task, the model is trained for 500,000 iterations with a batch size of 16 and a learning rate of 7×10^{-6}. It is finetuned on a single Tesla V100 GPU. We train the model with a mixture of imitation learning (IL) and A2C reinforcement learning (RL) [83]. The best model is selected according to performance on validation unseen split.

TABLE 3.2. Comparison with state-of-the-art methods on R2R.

Methods	Validation Seen				Validation Unseen				Test Unseen			
	TL	NE ↓	SR ↑	SPL ↑	TL	NE ↓	SR ↑	SPL↑	TL	NE ↓	SR ↑	SPL ↑
SF [24]	-	3.36	66	-	-	6.62	35	-	14.82	6.62	35	28
RCM [122]	10.65	3.53	67	-	11.46	6.09	43	-	11.97	6.12	43	38
Regretful [77]	-	3.23	69	63	-	5.32	50	41	13.69	5.69	48	40
Fast-short [49]	-	-	-	-	21.17	4.97	56	43	22.08	5.14	54	41
EnvDrop [116]	11.00	3.99	62	59	10.70	5.22	52	48	11.66	5.23	51	47
OAAM [94]	10.20	-	65	62	9.95	-	54	50	10.40	5.30	53	50
EntityGraph [36]	10.13	3.47	67	65	9.99	4.73	57	53	10.29	4.75	55	52
NvEM [2]	11.09	3.44	69	65	11.83	4.27	60	55	12.98	4.37	58	54
ActiveVLN[119]	19.70	3.20	70	52	20.6	4.36	58	40	21.6	4.33	60	41
Press [61]	10.57	4.39	58	55	10.36	5.28	49	45	10.77	5.49	49	45
PREVALENT [30]	10.32	3.67	69	65	10.19	4.71	58	53	10.51	5.30	54	51
DR-Attacker [66]	-	3.52	70	67	-	4.99	53	48	-	5.53	52	49
RecBERT [39]	11.13	2.90	72	68	12.01	3.93	63	57	12.35	4.09	63	57
AirBERT [29]	11.09	2.68	75	70	11.78	4.01	62	56	12.41	4.13	62	57
REM [69]	10.88	2.48	75	72	12.44	3.89	64	58	13.11	3.87	65	59
SOAT [84]	-	-	-	-	12.15	4.28	59	53	12.26	4.49	58	53
HAMT-ResNet152 [15]	-	-	69	65	-	-	64	58	-	-	-	-
HAM-ViT [15]	11.15	2.51	76	72	11.46	**2.29**	66	**61**	12.27	3.93	65	**60**
HOP [96]	11.26	2.72	75	70	12.27	3.80	64	57	12.68	3.83	64	59
HOP+	11.31	**2.33**	**78**	**73**	11.76	3.49	**67**	**61**	12.67	**3.71**	**66**	**60**

3.4.2 Quantitative Results

Room-to-Room (R2R)

Evaluation Metrics For the Room-to-Room task, we evaluate our model with four metrics:

- TL **Trajectory Length** measures the average length of all the predicted navigation trajectories in meters.
- NE **Navigation Error** is the mean of the shortest path distance in meters between the agent's final location and the target location.
- SR **Success Rate** measures the ratio of successful tasks, of which the agent's stop location is less than 3 meters away from the target location.
- SPL **Success weighted by Path Length [3]** trades-off SR (Success Rate) against TL (Trajectory Length). It measures both the accuracy and efficiency of navigation. SPL is the key metric for R2R.

Comparison with SoTA As shown in Table 3.2, our method achieves better performance compared to the state-of-the-art models, such as HAMT [15]. It is worth noting that HAMT was trained on 20 NVIDIA V100 GPUs, while our model uses only a quarter of its computational resources. With equal use of the ResNet152 feature, our approach outperforms HAMT in both val seen split (↑ 8% in SPL) and val unseen split (↑ 3% in SPL). Compared to our previous method HOP, our new method with a memory network outperforms them by 4% in val unseen split according to the main metric SPL, which indicates that our agent with an external memory network can effectively improve the navigation ability of the agent.

TABLE 3.3. Comparison with the state-of-the-art methods on REVERIE. SPL is the main metric for its navigation sub-task, and RGSPL is the main metric for the object grounding sub-task.

Methods	REVERIE Validation Seen						REVERIE Validation Unseen						REVERIE Test Unseen					
	Navigation				RGS↑	RGSPL↑	Navigation				RGS↑	RGSPL↑	Navigation				RGS↑	RGSPL↑
	SR↑	OSR↑	SPL↑	TL			SR↑	OSR↑	SPL↑	TL			SR↑	OSR↑	SPL↑	TL		
Human	–	–	–	–	–	–	–	–	–	–	–	–	81.51	86.83	53.66	21.18	77.84	51.44
RCM [122]	23.33	29.44	21.82	10.70	16.23	15.36	9.29	14.23	6.97	11.98	4.89	3.89	7.84	11.68	6.67	10.60	3.67	3.14
SMNA [76]	41.25	43.29	39.61	7.54	30.07	28.98	8.15	11.28	6.44	9.07	4.54	3.61	5.80	8.39	4.53	9.23	3.10	2.39
FAST-Short [49]	45.12	49.68	40.18	13.22	31.41	28.11	10.08	20.48	6.17	29.70	6.24	3.97	14.18	23.36	8.74	30.69	7.07	4.52
FAST-MATTN [95]	50.53	55.17	45.50	16.35	31.97	29.66	14.40	28.20	7.19	45.28	7.84	4.67	19.88	30.63	11.61	39.05	11.28	6.08
ORIST [93]	45.19	49.12	42.21	10.73	29.87	27.77	16.84	25.02	15.14	10.90	8.52	7.58	22.19	29.20	18.97	11.38	10.68	9.28
RecBERT [39]	51.79	53.90	47.96	13.44	38.23	35.61	30.67	35.02	24.90	16.78	18.77	15.27	29.61	32.91	23.99	15.86	16.50	13.51
AirBERT [29]	47.01	48.98	42.34	15.16	32.75	30.01	27.89	34.51	21.88	18.71	18.23	14.18	30.28	34.20	23.61	17.91	16.83	13.28
HAMT-ViT [15]	-	-	-	-	-	-	32.95	36.84	30.20	14.08	18.92	17.28	30.40	33.41	26.67	13.62	14.88	12.08
HOP [96]	53.76	54.88	47.19	13.80	38.65	33.85	31.78	36.24	26.11	16.46	18.85	15.73	30.17	33.06	24.34	16.38	17.69	14.34
HOP+	**55.87**	**56.43**	**49.55**	10.59	**40.76**	**36.22**	**36.07**	**40.04**	**31.13**	14.57	**22.49**	**19.33**	**33.82**	**35.81**	**28.24**	15.17	**20.20**	**16.86**

REVERIE

Evaluation Metrics REVERIE adopts five metrics: Success Rate (SR), Success weighted by Path Length (SPL), Oracle Success Rate (OSR), Remote Grounding Success rate (RGS), and RGS weighted by Path Length (RGSPL).

`OSR` **Oracle Success Rate** measures the ratio of tasks of which one of its trajectory viewpoints can observe the target object within 3 meters.

`RGS` **Remote Grounding Success rate** measures the ratio of tasks that successfully locate the target object.

`RGSPL` **RGS weighted by Path Length** is RGS weighted by Path Length, which is the main metric for this task.

Comparison with SoTA As shown in Table 3.3, our method outperforms previous approaches in both navigation performance (SR and SPL) and object grounding performance (RGS and RGSPL). Particularly, in the Test Unseen split, our method (HOP+) achieves 20.20% in RGS and 16.86% in RGSPL, which are significantly better than HAMT (↑ 5.32 in RGS and ↑ 4.78 in RGSPL). Considering our method just needs one-quarter of the resources of HAMT, our method is much more effective and efficient. We attribute this improvement to both the pre-training and the newly devised memory module as our previous HOP is already slightly better than HAMT and the memory module enlarges the superiority.

Room-Across-Room (RxR)

Evaluation Metrics The following four metrics are used for evaluation on the RxR dataset: Success Rate (SR), Success weighted by Path Length (SPL), normalized Dynamic Time Warping (nDTW) [45], and success rate weighted by Dynamic Time Warping (sDTW).

`nDTW` **Normalized Dynamic Time Warping** penalizes deviations from the reference path.

`sDTW` **Success weighted by normalized Dynamic TimeWarping** constrains nDTW to only successful episodes and effectively captures both success and fidelity.

Table 3.4. Comparison with state-of-the-art methods on RxR using English instructions.

Methods	RxR Validation Seen				RxR Validation Unseen			
	SR↑	SPL ↑	nDTW ↑	sDTW ↑	SR↑	SPL ↑	nDTW ↑	sDTW↑
Baseline [54]	28.6	-	0.45	0.23	26.1	-	0.42	0.21
EnvDrop [116]	48.1	44.0	0.57	0.40	38.5	34.0	0.51	0.32
EnvDrop+Syntax [60]	48.1	44.0	0.58	0.40	39.2	35.0	**0.52**	0.32
HOP [96]	49.4	45.3	0.58	0.40	42.3	36.3	**0.52**	0.33
HOP+	**53.6**	**47.9**	**0.59**	**0.43**	**45.7**	**38.4**	**0.52**	**0.36**

Table 3.5. Comparison with state-of-the-art methods on NDH measured by the Goal Progress in meters.

Methods	NDH Validation Unseen			NDH Test Unseen		
	Oracle	**Navigator**	**Mixed**	**Oracle**	**Navigator**	**Mixed**
Seq2Seq [117]	1.23	1.98	2.10	1.25	2.11	2.35
CMN [139]	2.68	2.28	2.97	2.69	2.26	2.95
PREVALENT [30]	2.58	2.99	3.15	1.67	2.39	2.44
ORIST [93]	3.30	3.29	3.55	2.78	3.17	3.15
DR-Attacker [66]	3.27	4.00	4.18	2.77	2.95	3.26
HOP [96]	4.07	4.05	4.41	2.99	3.18	3.24
HOP+	**4.32**	**4.21**	**4.59**	**3.25**	**3.60**	**3.90**

As shown in Table 3.4, our model achieves better performance than other SoTA methods on val seen and unseen sets. In particular, our model outperforms the previous SoTA method Syntax [60] on the unseen split (6.5% improvement in SR). Compared to our previous method HOP, the newly proposed method also improves almost all metrics, especially the success rate (3.4% improvement) in the unseen split. These results demonstrate the effectiveness of both the pre-training and the memory network for fine-tuning.

Navigation from Dialog History (NDH)

Evaluation Metric NDH uses Goal Progress (GP) in meters as the primary evaluation metric.

`GP` **Goal Progress** measures the average progress of the agent towards the target.

Comparison with SoTA Table 3.5 shows navigation results on the NDH task. Compared to the SoTA method DR-Attacker [66], HOP+ achieves much better results on both validation and test unseen environments in all settings. Specifically, on the validation unseen split under the mixed setting, our HOP+ model achieves up to 0.64 meter gains over DR-Attacker. In addition, HOP+ significantly outperforms PREVALENT (by 1.46 meter) on the Test split under the Oracle setting, which is the pre-training method. Compared to our previous method HOP, our new method also achieves better results on all splits under all settings (with 0.16∼0.66 meters improvements), which indicates the effectiveness of the memory module.

TABLE 3.6. Ablation study of the pre-training tasks on validation unseen splits of three VLN tasks: R2R, REVERIE, and NDH.

	Pre-training Tasks						R2R		REVERIE					NDH
	MLM	TIM	TOM	GOM	APH	AP	SR↑	SPL↑	SR↑	OSR↑	SPL↑	RGS↑	RGSPL↑	Goal Progress↑
1							55.90	50.55	25.11	28.71	20.46	15.42	12.70	3.42
2	✓						61.45	55.42	25.90	28.96	21.70	16.47	13.85	3.82
3	✓	✓					61.77	55.95	27.92	32.12	22.27	17.98	14.37	3.89
4	✓		✓				62.74	56.36	28.88	32.01	23.89	19.20	15.93	3.92
5	✓			✓			63.25	56.89	29.34	32.72	24.82	18.46	15.80	3.94
6	✓				✓		64.82	58.91	33.79	35.90	28.28	20.73	17.68	4.19
7	✓					✓	64.11	57.37	33.03	36.98	28.16	20.13	16.85	4.06
8	✓	✓	✓	✓			65.86	58.48	33.43	37.26	28.62	20.99	17.98	4.31
9	✓	✓	✓	✓	✓		**66.41**	**60.20**	**34.14**	**37.52**	**29.37**	**21.61**	**18.56**	**4.45**

3.4.3 Ablation Study

The Effect of proxy tasks We analyze the advantage of different pre-training tasks through ablation studies over R2R, REVERIE, and NDH downstream tasks. As shown in Table 3.6, the first row trains HOP+ from scratch with a memory network on the selected VLN tasks. Row 2 shows the results from pre-training with MLM task, where we can see a gain with large margins on all VLN tasks, especially with approximately +6 SR and +5 SPL on the R2R task. While Row 3∼Row 9 show the result of incorporating MLM task with other proxy tasks of TIM, TOM, GOM, APH, and AP. The result shows combining APH with MLM brings the largest improvement compared to other single proxy tasks, which illustrates the superiority of our proposed Action Prediction with the History proxy task. Furthermore, it surpasses AP task which directly predicts actions without history trajectory by non-trivial margins. In other words, both action prediction and history knowledge play vital roles in pre-training for VLN downstream tasks. While either combing the order modeling proxy task of TOM or GOM with MLM can outperform combing TIM with MLM, showing both these two tasks can learn a better visiolinguistic representation than the widely used trajectory-instruction matching task. When assembling these proxy tasks, the performances are further improved as shown in Row 8 and Row 9. Compared to the baseline model without pre-training, a great performance improvement of 10.5% SR on R2R, 9% SR on REVERIE, and 1 meter on NDH is achieved.

From the view of downstream tasks, on the most complicated REVERIE task that not only measures the agent's ability in navigation but also in object grounding, HOP+ achieves the most obvious improvement with a relative 43.5% on SPL and 46.1% on RGSPL compared to the baseline model. While on another comprehensive NDH task where the instructions are ambiguous and implied in historical dialogues and historical visual trajectories, HOP+ improves the performance with a relative gain of 30.1%. In fact, our proposed proxy tasks and the memory network are exactly suitable for handling such two tasks, which significantly boosts navigation performance. All these promising results show the effectiveness of our proposed HOP+.

TABLE 3.7. Ablation study results of pre-training data on the validation unseen splits of R2R, REVERIE, and RxR tasks. BnB* denotes our processed data from AirBERT [29].

Pre-training Data	R2R		REVERIE		RxR	
	SR↑	SPL↑	SR↑	SPL↑	SR↑	SPL↑
None	55.90	50.55	25.11	20.46	37.8	32.5
PREVALENT	66.41	60.20	34.14	29.37	45.0	37.9
PREVALENT + BnB*	**66.84**	**60.94**	**36.07**	**31.13**	**45.7**	**38.4**

The Effect of Pre-training Data

To evaluate the effect of different pre-training data, we conduct an ablation study on R2R, REVERIE, and RxR validation unseen split. As shown in Table 3.7, when pre-trained with data from PREVALENT, our model significantly improves performance on all these tasks (8%∼11%). And the performance is still able be improved when adding data processed from AirBERT (denoted as BnB* in the table). In addition, we observe that the BnB* data benefits the REVERIE task the most. This may be due to the BnB's captions mainly describing rooms and objects, which matches REVERIE's object grounding mission. Our results also indicate that more data is usually helpful but the marginal utility will also become obvious.

History Information in Pre-training

Figure 3.11 presents the result of the Action Prediction task whether using historical information in pre-training. It shows that the validation curves of our Action Prediction with History(APH) converges faster and with better accuracy than Action Prediction (AP) task during pre-training. Moreover, the results illustrated in Table 3.6 (Model 6 and Model 7) also show that APH achieves higher performance for all metrics and on three downstream tasks. This evaluation demonstrates the advantage of our Action Prediction with History tasks compared to the normal Action Prediction task.

APH vs APM in Fine-tuning

Here we compare two kinds of approaches that handle history information: 1) Action Prediction with history (APH): we consider historical visual observations the same as the pre-training task APH to aid action prediction. 2) Action Prediction with memory (APM): our proposed external memory network that learns to select useful information from the historical trajectory. We use HOP [96] as a baseline, which does not utilize history information as APH or APM. We conduct experiments on the validation split of the R2R task. As shown in Figure 3.12, when introducing historical information, the agent's navigation performance improves on both seen and unseen splits. When we additionally utilize an external memory network, the performance is

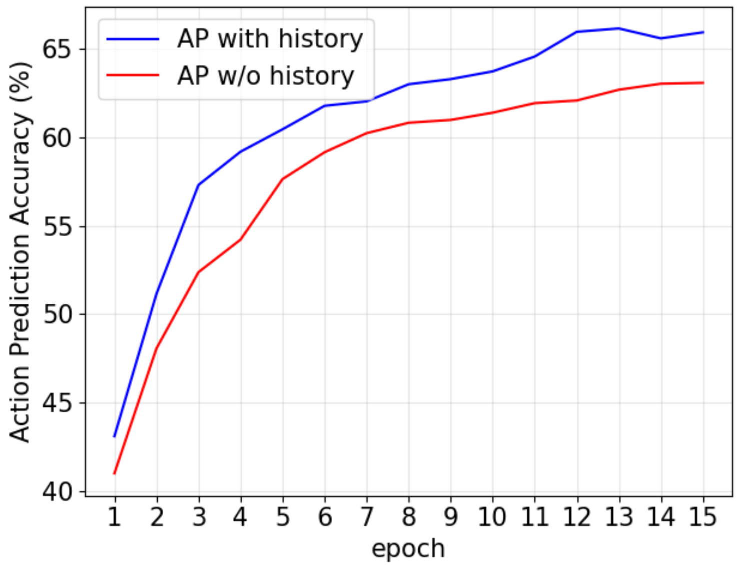
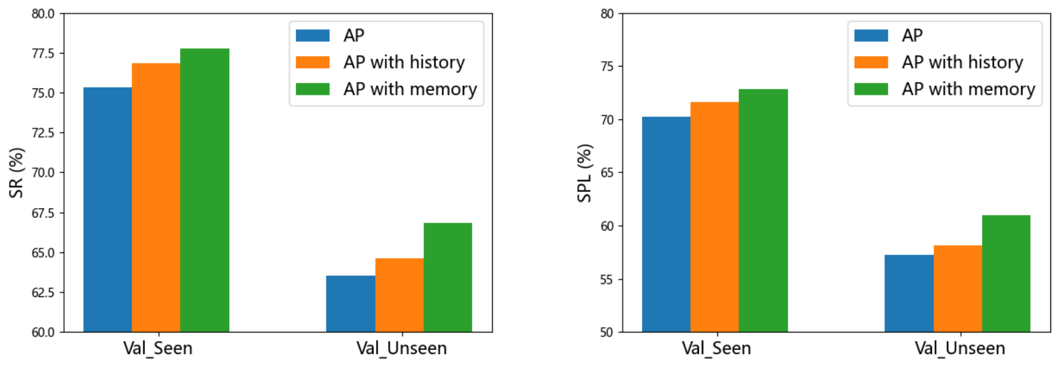


FIGURE 3.11. APH *v.s.* AP regarding action prediction accuracy.



(A) SR(%)

(B) SPL(%)

FIGURE 3.12. Effect of memory network in fine-tuning.

further significantly improved, especially on the unseen split (↑ 2.22% in SR, ↑ 2.81% in SPL). This suggests that the introduction of an additional attention mechanism can indeed help the agent in action decision-making.

Computational Efficiency

To assess the computational efficiency of the way of utilizing historical information, we compare the inference time of AP, APM, and APH, where AP represents the original HOP which does not handle history information in fine-tuning and inference. We run each model on the R2R val unseen split (2349 instructions) using a single 1080Ti GPU. We run each model three times and calculate the average inference time. As shown in Table 3.8, we can see that the inference time using APM increases 9.3% compared to AP. While APH increases much more inference time, with a large margin of 28.7%

Table 3.8. Inference time on R2R validation unseen split.

Methods	AP	APH	APM
Inference time (s)	43.58	56.08	47.65

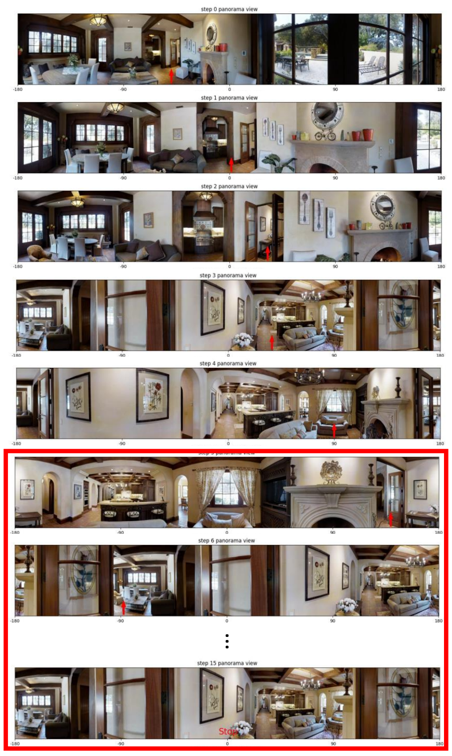
compared to AP. The increased inference time of APH is about 3 times to that of APM.

3.4.4 Qualitative results

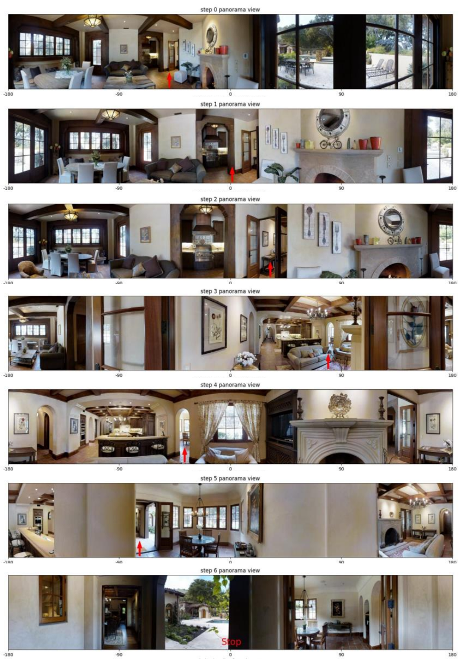
Figure 3.13 and Figure 3.14 respectively visualize trajectories predicated by our method HOP+ compared to our preliminary version HOP [96] and RecBERT [39]. As shown in Figure 3.13, we can see that HOP could follow the instruction in the first few steps, and as the number of steps increases, it fails to reach the target location. By contrast, our HOP+ enables the model to recognize the scene such as the kitchen, living room, and dining room, then well follow the instruction to stop on the porch. As shown in Figure 3.14, RecBERT fails from the first step, where it cannot find the vase mentioned in the instruction. Although it later finds a bed, it is found in the wrong room. By contrast, our HOP+ recognises the vase and then well follow the instruction to stop at the right bathroom. We also provide a fail case of our method in Figure 3.15. When going through the hall, there are two doorways. HOP+ chooses the wrong direction when crossing the hall (it does not choose the doorway directly opposite). This indicates that HOP+ fails to understand orientation words such as “opposite” and that the learning ability of language representation which has not received much attention should be enhanced in future work.

3.5 Conclusion

In this work, we introduce a history-enhanced and order-aware pre-training with the complementing fine-tuning paradigm for Vision-and-Language Navigation tasks. We design three tailored pre-training tasks for VLN: Action Prediction with History (APH), Trajectory Order Modeling (TOM), and Group Order Modeling (GOM). These tasks equip the agent to be aware of history information and temporal order within instructions. In addition, to bridge the gap between the pre-training and fine-tuning stages, we propose an external memory network in finetuning stage to effectively utilize the historical knowledge learned during pre-training and enhance the agent’s ability in decision-making. We conduct extensive experiments on four mainstream downstream VLN tasks, including R2R, REVERIE, RxR, and NDH, and the results demonstrate the effectiveness of our proposed HOP+.

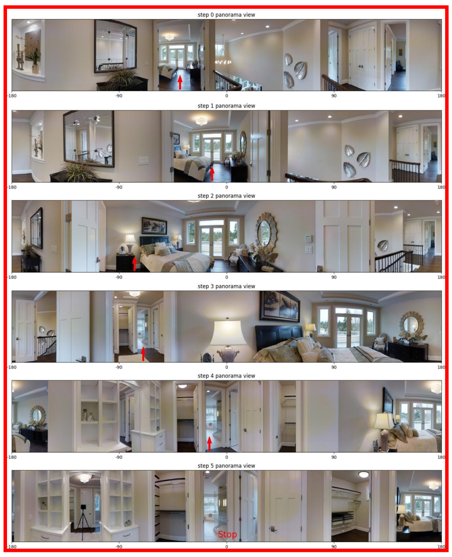


(A) Predicted trajectory by HOP [96] (failed).

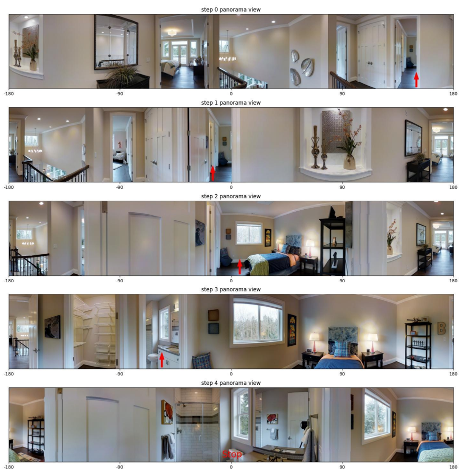


(B) Predicted trajectory by HOP+ (succeed).

Figure 3.13. Comparisons of predicted trajectory with our previous work HOP [96]. The navigation steps inside the red box are incorrect. Instruction:" leave sitting room and head towards the kitchen, turn right at living room and enter. walk through living room to dinning room and enter. Turn left and head to front door. Exit the house and stop on porch."

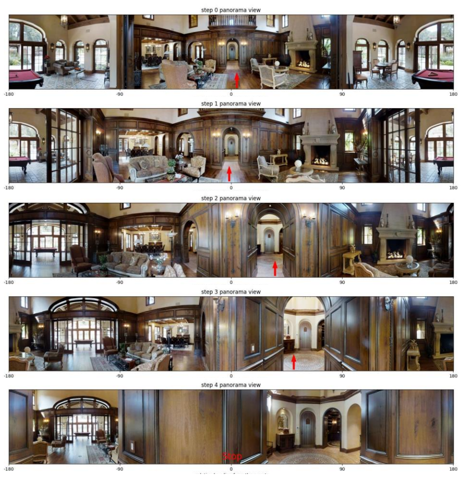


(A) Predicted trajectory by RecBERT [39] (failed).

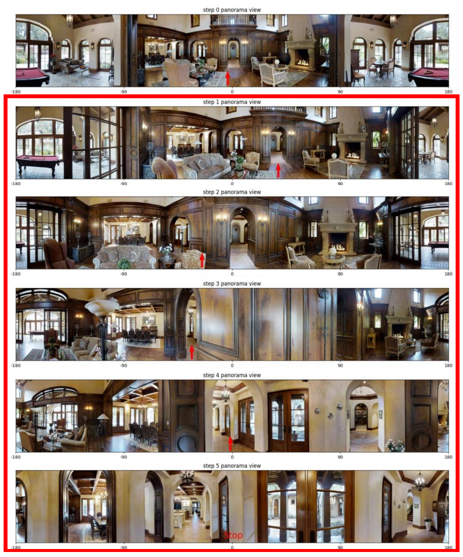


(B) Predicted trajectory by HOP+ (succeed)

FIGURE 3.14. Comparisons of predicted trajectory with state-of-the-art method RecBERT [22]. Navigation steps inside red box are incorrect. Instruction: "Walk down the hallway past the vases, enter the bedroom, walk to the foot of the bed, walk toward the bathroom on the left, wait in the doorway to the bathroom."



(A) Ground-truth trajectory.



(B) Predicted trajectory by HOP+ (failed)

FIGURE 3.15. Fail case of HOP+ in R2R val unseen split. Instruction: "Move forward to the doorway on the opposite side of the hall. Stop in the archway."

Statement of Authorship

Title of Paper	HOP: History-and-Order Aware Pre-training for Vision-and-Language Navigation.
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication; Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in CVPR 2022, 15397-15406

Principal Author

Name of Principal Author (Candidate)	Yanyuan Qiao		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	28/06/2023

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yuankai Qi		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Yicong Hong		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Zheng Yu

Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Peng Wang		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Qi Wu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	29/06/2023

Statement of Authorship

Title of Paper	HOP+: History-enhanced and Order-aware Pre-training for Vision-and-Language Navigation.
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2023, 8524-8537

Principal Author

Name of Principal Author (Candidate)	Yanyuan Qiao		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	28/06/2023

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yuankai Qi		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Yicong Hong		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Zheng Yu

Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Peng Wang		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Qi Wu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	29/06/2023

Chapter 4

Parameter-Efficient Transfer Learning

Though Chapter 3 addressed a History-enhanced and Order-Aware Pre-training with the complementing fine-tuning paradigm to solve the current lack of lacking a pre-trained model that emphasizes temporal and historical information in VLN tasks, there still exists a challenge that the inefficient problem of tuning large pre-trained models for VLN tasks. Thus, in this chapter, we present the first study of applying Parameter-Efficient Transfer Learning (PETL) techniques to VLN tasks and propose a VLN-specific PETL method named VLN-PETL.

4.1 Introduction

Large-scale pre-trained models have shown remarkable success in both computer vision (CV) and natural language processing (NLP) domains, and have largely improved the performance of a variety of visio-linguistic tasks [57, 87, 129]. These models follow a standard pretrain-and-finetune paradigm, which first pretrains the model on large-scale unlabeled data and then finetunes it on each downstream task. Since the size of such models is growing rapidly [97], even fully finetuning and storing a copy of the entire pretrained model for each downstream becomes costly.

To alleviate this problem, Parameter-Efficient Transfer Learning (PETL) has been proposed as an alternative training strategy [7, 34, 40, 41, 78, 79] and initially achieved great progress in NLP community. These methods aim to exploit the representation knowledge in the large pretrained models by freezing most parameters of the model and only tuning a small set of parameters, which can achieve comparable or even better performance to full fine-tuning. Several approaches have attempted to apply PETL techniques to CV and V&L domains [35, 88, 112, 127] and achieved promising results on various downstream tasks. Recent works [31, 81, 132] find that different PETL methods have different characteristics and performance on the same downstream task and thus combining multiple PETL techniques may be more effective in improving the performance.

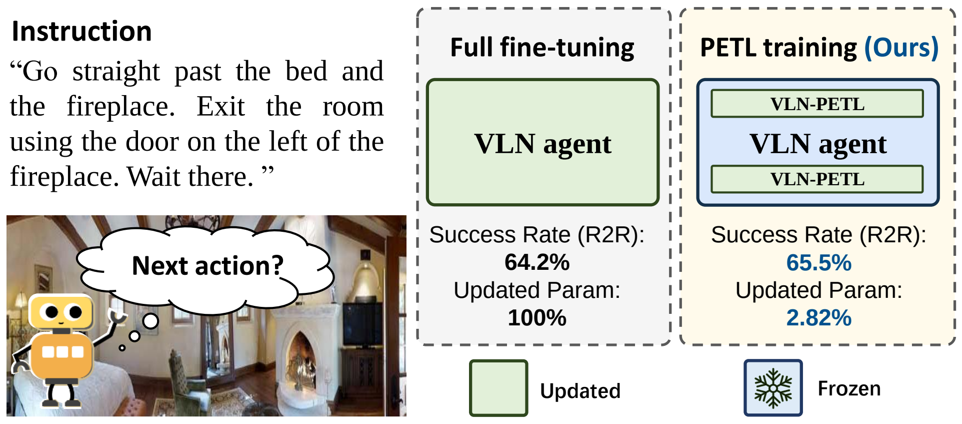

Instruction
"Go straight past the bed and the fireplace. Exit the room using the door on the left of the fireplace. Wait there. "
Next action?
Full fine-tuning
VLN agent
Success Rate (R2R): 64.2%
Updated Param: 100%
PETL training (Ours)
VLN-PETL
VLN agent
VLN-PETL
Success Rate (R2R): 65.5%
Updated Param: 2.82%
Updated
Frozen

FIGURE 4.1. Comparison of full fine-tuning and our proposed PETL training for VLN tasks. By updating only a small subset of parameters, our proposed VLN-PETL can achieve a comparative performance compared to full fine-tuning.)

Vision-and-Language Navigation (VLN), which deals with visual, linguistic, and robotic action inputs simultaneously, could benefit from the pre-trained large models while suffering from the considerable model size during the downstream tasks fine-tuning. Considering downstream VLN agents are complex enough, full finetuning them with the large pre-trained models for each downstream VLN task becomes expensive, in which case the technique of PETL shows great potential. Unlike most NLP, CV, and V&L tasks, VLN is a dynamic action decision-making task relying on the current environment and previous history knowledge of the chosen actions. Specifically, given the instruction in natural language, the VLN agent perceives a new visual observation according to the chosen action at the previous timestep and should choose the next action to perform at the current timestep. Thus, how to effectively learn history knowledge is crucial to adapting PETL methods for VLN tasks. Moreover, the cross-modal interaction which plays a vital role in action prediction should be also enhanced during the process of efficient tuning. Besides, our experiments show that directly applying some existing PETL methods to VLN tasks may bring non-trivial performance degeneration.

Considering these reasons, we propose a VLN-specific PETL method named VLN-PETL. Specifically, we design two tailored PETL modules for VLN: Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). Both these two modules mainly consist of bottleneck layers and multi-head cross-attention layers. HIB enhances the interaction between the observation and the previous historical knowledge in a recurrent pattern. While CIB adopts a two-stream structure to focus on the interaction of cross-modal knowledge. Similar to adapters that inject bottleneck layers into transformer blocks for efficient tuning, we insert HIB and CIB into the visual encoder and cross-modal encoder separately in the pre-trained model for VLN. During the training process, the original weights of the large pre-trained model are frozen and only weights of these newly injected modules are trained and updated for different downstream VLN tasks.

In addition to HIB and CIB, VLN-PETL also adopts vanilla adapters to efficiently tune the language encoder and the LoRA to further improve the parameter-efficient tuning's performance as previous work declared [31, 81, 132] for downstream VLN tasks.

We conduct extensive experiments on four mainstream VLN tasks: R2R [5], REVERIE [95], NDH [117], and RxR [54]. The results show that VLN-PETL not only surpasses other PETL methods with promising margins but also achieves comparable or even better performance compared to full fine-tuning, especially on R2R ($\uparrow$ 1.3% SR on validation unseen set, updating only 2.82% params, see Figure 4.3) and on NDH ($\uparrow$ 1.08 GP on test unseen set which achieves the top position in the leaderboard). We also conduct ablation studies to evaluate the contribution of each component of VLN-PETL and validate the superiority of HIB and CIB to counterpart PETL methods.

In summary, our contributions are as follows: (1) We present the first study that explores Parameter-Efficient Transfer Learning (PETL) techniques for Vision-and-Language Navigation (VLN) tasks; (2) We propose a VLN-specific PETL method named VLN-PETL, which incorporates existing PETL methods with two tailored PETL modules for VLN tasks: Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB); (3) Extensive experiments on four VLN downstream tasks demonstrate the effectiveness of our proposed VLN-PETL, which outperforms other PETL methods and keep competitive to full fine-tuning with much fewer trainable parameters.

4.2 Background

Parameter-Efficient Transfer Learning Methods

Recently, with the rapid increase of pre-trained models' size, how to efficiently tune the large pre-trained models has received great attention and Parameter-Efficient Transfer Learning (PETL) has become a popular research area. One category of PETL methods adds new parameters into the pretrained model and only trains these parameters. For example, Adapter [40] introduces bottleneck layers after attention layers and feed-forward layers in the transformer block. LoRA [41] injects trainable low-rank decomposition matrices into linear projection layers to approximate the update of large amount of parameters. Prompt Tuning [58] prepends trainable prompts to the model's input. Another kind of PETL method does not add new parameters and selects a subset of the pretrained model's parameters to update, such as BitFit [7], which only trains the bias term in the model. Recent works [31, 81, 132] find that incorporating different PETL methods as sub-modules may help improve the integrated performance for different downstream tasks in NLP and CV.

However, VLN is a dynamic task of action prediction relying on both the current observations and previous action decisions, which is more challenging than other static NLP, CV and V&L tasks. In other words, the previous decision of action will influence the current environment observed by the agent as well as the choice of the next action to perform. Thus, it is important to effectively utilize the historical knowledge of the previous trajectory when applying PETL methods to VLN. Furthermore, the cross-modal interactions of the language and vision also have a great impact on action predictions, which should be enhanced especially when most parameters of the pre-trained model are frozen for efficient tuning. Therefore, we specifically design two PETL modules for VLN tasks to enhance the history knowledge interactions and cross-modal knowledge interactions, namely Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). In addition, VLN-PETL also incorporates vanilla adapters to efficiently tune the language encoder and LoRA to further improve the performance.

Here we give descriptions about PETL methods:

Adapter inserts trainable bottleneck layers after the multi-head attention layers or feed-forward layers in the transformer blocks. The bottleneck layer of an adapter consists of a linear down-projection with $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{D_{\text{hidden}} \times D_{\text{mid}}}$, a non-linear activation function $\sigma(\cdot)$ and a linear up-projection $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{D_{\text{mid}} \times D_{\text{hidden}}}$. Given the input feature $\boldsymbol{f}_{\text{in}}$, the adapter first projects $\boldsymbol{f}_{\text{in}}$ into the D_{mid} bottleneck dimension and then recovers it back into D_{hidden} dimension as:

$$\boldsymbol{f}_{\text{out}} = \boldsymbol{W}_{\text{up}}^{\intercal} \sigma(\boldsymbol{W}_{\text{down}}^{\intercal} \boldsymbol{f}_{\text{in}}). \tag{4.1}$$

Bias terms are omitted for brevity. The parameters of layer normalization are usually tuned together with the adapter. Besides, the adapter can be inserted into the transformer layers in a sequential manner or in a parallel manner, while the latter one is proved superior to the former one as in [31].

LoRA injects trainable low-rank decomposition matrices to represent the weight updates of the frozen parameters in the transformer's linear projection layers. Specifically, for a weight matrix $\boldsymbol{W} \in \mathbb{R}^{D_{\text{hidden}} \times D_{\text{hidden}}}$ in the pre-trained model, the weight update $\Delta \boldsymbol{W} \in \mathbb{R}^{D_{\text{hidden}} \times D_{\text{hidden}}}$ is approximated by two low-rank matrices $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{D_{\text{hidden}} \times D_{\text{mid}}}$ and $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{D_{\text{mid}} \times D_{\text{hidden}}}$ as follow:

$$\boldsymbol{W} + \Delta \boldsymbol{W} = \boldsymbol{W} + \boldsymbol{W}_{\text{down}} \boldsymbol{W}_{\text{up}}, \tag{4.2}$$

and the forward pass of LoRA can be formulated as:

$$\boldsymbol{f}_{\text{out}} = (\boldsymbol{W}^{\intercal} + \gamma \boldsymbol{W}_{\text{up}}^{\intercal} \boldsymbol{W}_{\text{down}}^{\intercal}) \boldsymbol{f}_{\text{in}}. \tag{4.3}$$

where γ is a fixed scalar hyperparameter for scaling.

Prompt Tuning prepends a sequence of randomly initialized continuous prompts $\boldsymbol{f}_{\text{prompt}}$ into the input feature $\boldsymbol{f}_{\text{in}}$. During training, only these prompts are optimized by updating a learnable projection matrix $\boldsymbol{W}_{\text{prompt}} \in \mathbb{R}^{1\times D_{\text{hidden}}}$. The forward pass can be formulated as:

$$\boldsymbol{f}_{\text{prompt}} = \boldsymbol{W}_{\text{prompt}}^{\intercal}\boldsymbol{X}, \tag{4.4}$$

$$\boldsymbol{f}_{\text{out}} = \text{TRM}([\boldsymbol{f}_{\text{prompt}}, \boldsymbol{f}_{\text{in}}]). \tag{4.5}$$

where $\boldsymbol{X}$ represents discrete prompt tokens, $[\cdot]$ represents concatenation, $\text{TRM}(\cdot)$ represents the transformer block.

BitFit does not introduce new parameters or inputs for tuning. It only tunes the bias terms in the pretrained models, which shows promising performance in some NLP tasks.

4.3 Method

We use a large pre-trained model of HAMT as our baseline, which is pre-trained on plentiful text-image pairs of instructions and the corresponding trajectories. As illustrated in Figure 4.2, HAMT adopts a two-stream architecture, which consists of a language encoder and a vision encoder to extract single-modal features and a cross-modal encoder to fuse multi-modal features for action prediction.

4.3.1 Baseline model

Language Encoder

Following the practice of BERT, the language encoder first embeds the instruction $\mathcal{I}$ into language embedding E_x by summing the word embedding, position embedding, and type embedding of each word x_i in the instruction. Then, E_x is passed through N_L transformer blocks which consist of a multi-head self-attention layer and feed-forward layer to generate the language feature $\boldsymbol{f}_x$.

Vision Encoder

The vision encoder mainly consists of observation encoding and history encoding. At time step t, the panoramic image feature and the corresponding angle feature a_t^o are projected into E_v^o and E_a^o followed by the layer normalization. Then, E_v^o and E_a^o are summed up as the current observation feature $\boldsymbol{o}_t$. Meanwhile, the panoramic image feature v_{t-1}^h and the corresponding turned angle feature a_{t-1}^h of the previous time step are taken as the input for history encoding. Similarly, v_{t-1}^h and a_{t-1}^h are first projected into E_v^h and E_a^h. Then, E_v^h and E_a^h are summed up and passed through N_H transformer blocks to generate h_t, which is appended into the tail of $\langle h_1, ..., h_{t-1}\rangle$

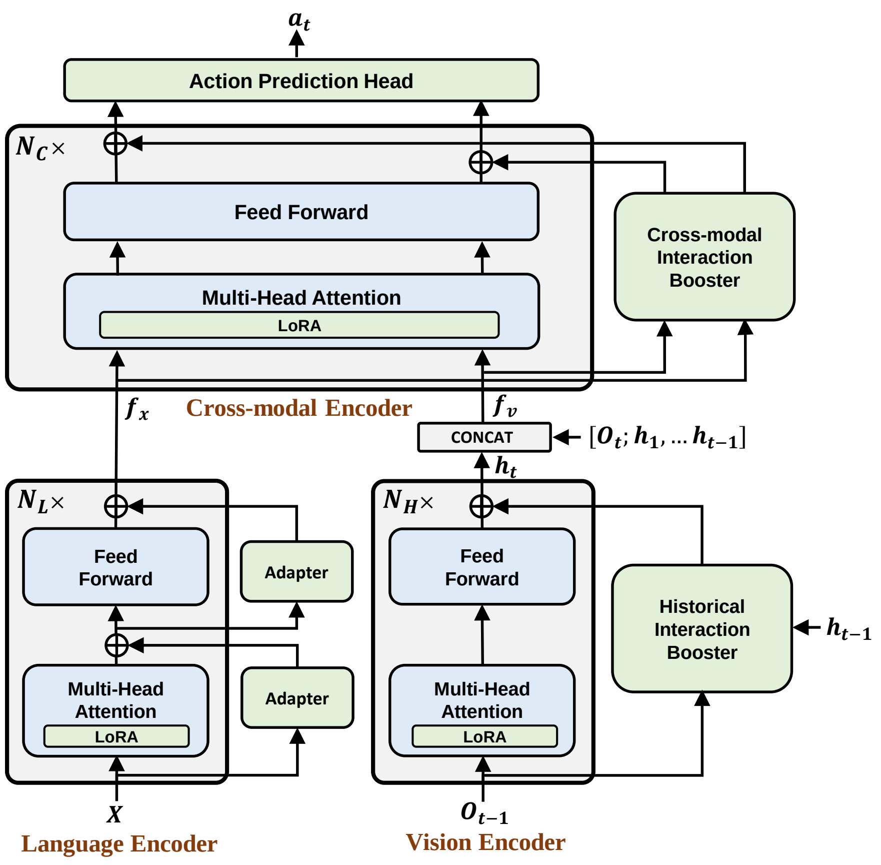


FIGURE 4.2. Illustration of the framework of our proposed VLN-PETL. The pre-trained model of HAMT mainly consists of a language encoder, a vision encoder, and a cross-modal encoder. The blue color denotes the frozen parameters in the pre-trained model and the green color denotes the trainable parameters of injected PETL modules.

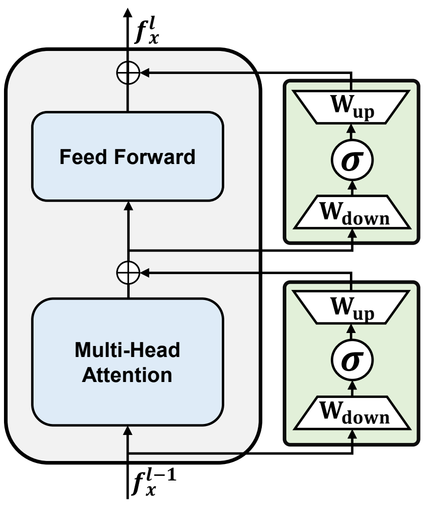
as the current time step's history feature $\boldsymbol{h}_t$. Indeed, the history encoding only re-encodes the previous time step's observation statically without any other interactions. Thus, this manner of history feature's generation and update may neglect the temporal knowledge embodied in the navigation trajectory.

Cross-modal Encoder

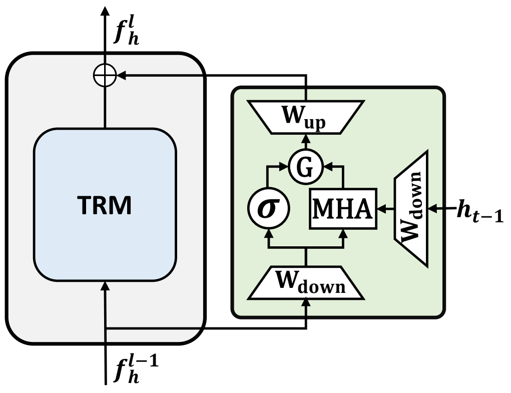
At the time step t, the observation feature $\boldsymbol{o}_t$ and the history feature $\boldsymbol{h}_t$ are first concatenated as the visual feature $\boldsymbol{f}_v$. Then, $\boldsymbol{f}_x$ and $\boldsymbol{f}_v$ are passed through N_C cross-modal transformer blocks that consist of two-stream multi-head cross-attention layers, multi-head self-attention layers, and feed-forward layers to generate cross-modal features for action prediction.

4.3.2 VLN-PETL

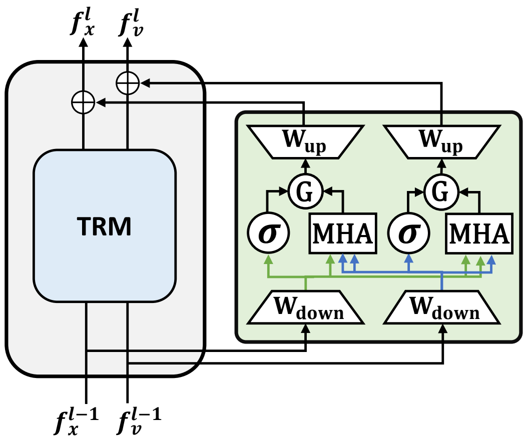
Not all the aforementioned PETL methods perform well in the complicated VLN tasks, such as BitFit and Prompt Tuning, which bring performance degeneration to VLN



(A) Language Encoder Adapter (LEA).



(B) Historical Interaction Booster(HIB).



(C) Cross-modal Interaction Booster(CIB).

FIGURE 4.3. Detailed components of VLN-PETL. TRM represents the transformer block, MHA represents the multi-head attention layer, σ represents the activation layer and G represents the learnable gate.

tasks (see Sec. 4.5 for evaluation and explanation). Thus, as shown in Figure 4.2, we only incorporate the adapter and LoRA as the PETL components of our integrated VLN-PETL. Furthermore, two block-level PETL modules are specially designed for VLN tasks considering the unique characteristics of VLN tasks and parameter efficiency, namely Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). Based on the bottleneck structure of the adapter, these two modules respectively strengthen the historical interaction and cross-modal interaction by incorporating the multi-head cross-attention mechanism and gating mechanism.

Language Encoder Adapter

As shown in Figure 4.3a, the Language Encoder Adapter (LEA) is inserted into the multi-head self-attention layers and feed-forward layers in parallel. Concretely, for the l-th transformer block in the language encoder, the input feature f_x^{l-1} is first passed through the adapter and summed with the output feature of the multi-head self-attention layer as $\boldsymbol{f}_{\text{att}}$:

$$\boldsymbol{f}_{\text{att}} = \text{LN}\big(\text{MSA}(\boldsymbol{f}_x^{l-1}) + \text{ADAPTER}(\boldsymbol{f}_x^{l-1})\big), \tag{4.6}$$

where MSA$(\cdot)$ represents the multi-head self-attention layer, ADAPTER$(\cdot)$ represents the adapter block shown in Eq.4.1 and LN$(\cdot)$ represents the layer normalization. Similarly, another adapter is inserted into the feed-forward layer which takes $\boldsymbol{f}_{\text{att}}$ as input:

$$\boldsymbol{f}_{\text{ffn}} = \text{LN}\big(\text{FFN}(\boldsymbol{f}_{\text{att}}) + \text{ADAPTER}(\boldsymbol{f}_{\text{att}})\big), \tag{4.7}$$

where FFN$(\cdot)$ represents the feed-forward layer. $\boldsymbol{f}_{\text{ffn}}$ is used as the final output feature $\boldsymbol{f}_x^l$ of the l-th transformer block:

$$\boldsymbol{f}_x^l = \boldsymbol{f}_{\text{ffn}}. \tag{4.8}$$

Historical Interaction Booster

As shown in Figure 4.3b, the Historical Interaction Booster (HIB) adopts the multi-head cross-attention mechanism to enhance the historical interaction between the observation feature and the history feature at each timestep t in a recurrent pattern. Specifically, for the l-th transformer block in the vision encoder, the input observation feature $\boldsymbol{f}_h^{l-1}$ and the previous history feature $\boldsymbol{h}_{t-1}$ are first downsampled into $\boldsymbol{f}_{\text{down}}$ and $\boldsymbol{h}_{\text{down}}$ with D_{mid} dimension by projection matrices $\boldsymbol{W}_{\text{down\_f}}$ and $\boldsymbol{W}_{\text{down\_h}}$:

$$\boldsymbol{f}_{\text{down}} = \boldsymbol{W}_{\text{down\_f}}^{\intercal} \boldsymbol{f}_h^{l-1}, \tag{4.9}$$

$$\boldsymbol{h}_{\text{down}} = \boldsymbol{W}_{\text{down\_h}}^{\intercal} \boldsymbol{h}_{t-1}, \tag{4.10}$$

Then, the history knowledge is encoded into the observation feature by the multi-head cross-attention between $\boldsymbol{f}_{\text{down}}$ and $\boldsymbol{h}_{\text{down}}$ followed by a learnable gate α:

$$\boldsymbol{f}'_{\text{down}} = \text{ReLU}(\boldsymbol{f}_{\text{down}}), \tag{4.11}$$

$$\boldsymbol{f}_{\text{cross}} = \text{MHA}(\boldsymbol{f}_{\text{down}}, \boldsymbol{h}_{\text{down}}), \tag{4.12}$$

$$\alpha = \text{Sigmoid}(\frac{\theta}{T}), \tag{4.13}$$

$$\boldsymbol{f}_{\text{v\_h}} = \alpha * \boldsymbol{f}'_{\text{down}} + (1-\alpha) * \boldsymbol{f}_{\text{cross}}, \tag{4.14}$$

where MHA$(\cdot)$ represents the multi-head cross-attention layer of which the query is $\boldsymbol{f}_{\text{down}}$ while the key and value are $\boldsymbol{h}_{\text{down}}$, θ is a learnable scalar initialized by zero and T is fixed as 0.1 representing the temperature hyperparameter. Next, the attended visual-and-historical feature $\boldsymbol{f}_{\text{v\_h}}$ is upsampled into D_{hidden} dimension and summed

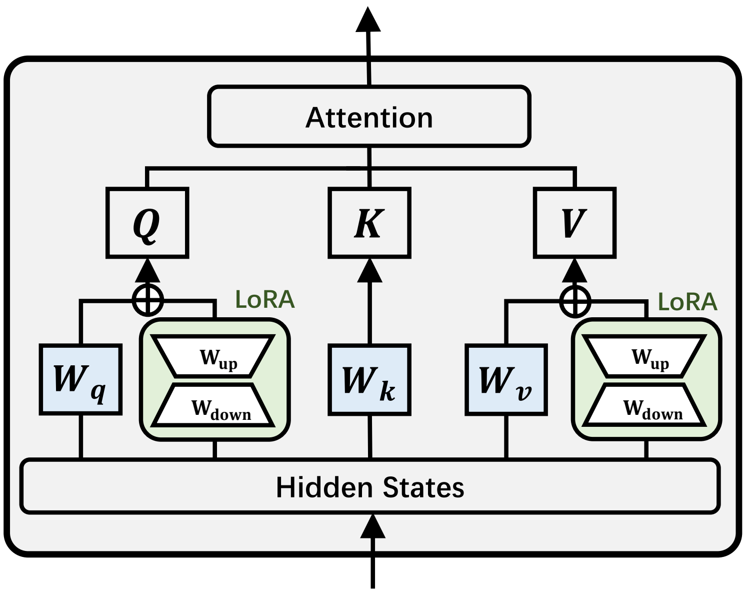


FIGURE 4.4. Illustration of injecting LoRA in the Multi-head Attention layer.

with the original output feature $\hat{\boldsymbol{f}}_t^l$ of the l-th transformer block:

$$\hat{\boldsymbol{f}}_h^l = \mathrm{TRM}(\boldsymbol{f}_h^{l-1}), \tag{4.15}$$

$$\boldsymbol{f}_h^l = \mathrm{LN}(\hat{\boldsymbol{f}}_h^l + \boldsymbol{W}_{\mathrm{up}}^{\intercal} \boldsymbol{f}_{\mathrm{v\_h}}), \tag{4.16}$$

where TRM$(\cdot)$ represents the transformer block in the vision encoder. The final output history feature h_t at the current timestep t is obtained as follow:

$$h_t = \boldsymbol{f}_h^L, \tag{4.17}$$

where L represents the number of transformer blocks.

Cross-modal Interaction Booster

As shown in Figure 4.3c, to enhance the interaction between language and visual modalities, Cross-modal Interaction Booster (CIB) adopts a two-stream multi-head cross-attention mechanism. To be specific, for the l-th transformer block in the cross-modal encoder, the input language feature $\boldsymbol{f}_x^{l-1}$ and the visual feature $\boldsymbol{f}_v^{l-1}$ are first downsampled into $\boldsymbol{f}_{\mathrm{down\_x}}$ and $\boldsymbol{f}_{\mathrm{down\_v}}$ with D_{mid} dimension by projection matrices $\boldsymbol{W}_{\mathrm{down\_x}}$ and $\boldsymbol{W}_{\mathrm{down\_v}}$ as Eq.4.9 and Eq.4.10. Then, a two-stream multi-head cross-attention is implemented by exchanging the query for the key and value as follows:

$$\boldsymbol{f}_{\mathrm{cross\_x}} = \mathrm{MHA}(\boldsymbol{f}_{\mathrm{down\_x}}, \boldsymbol{f}_{\mathrm{down\_v}}), \tag{4.18}$$

$$\boldsymbol{f}_{\mathrm{cross\_v}} = \mathrm{MHA}(\boldsymbol{f}_{\mathrm{down\_v}}, \boldsymbol{f}_{\mathrm{down\_x}}), \tag{4.19}$$

Two learnable gates α_x and α_v are used to obtain cross-attended language feature $\boldsymbol{f}_{\mathrm{x\_v}}$ and visual feature $\boldsymbol{f}_{\mathrm{v\_x}}$:

$$\boldsymbol{f}_{\mathrm{x\_v}} = \alpha_x * \boldsymbol{f}'_{\mathrm{down\_x}} + (1-\alpha_x) * \boldsymbol{f}_{\mathrm{cross\_x}}, \tag{4.20}$$

$$\boldsymbol{f}_{\mathrm{v\_x}} = \alpha_v * \boldsymbol{f}'_{\mathrm{down\_v}} + (1-\alpha_v) * \boldsymbol{f}_{\mathrm{cross\_v}}, \tag{4.21}$$

where $\boldsymbol{f}'_{\mathrm{down\_x}}$ and $\boldsymbol{f}'_{\mathrm{down\_v}}$ are obtained by passing $\boldsymbol{f}_{\mathrm{down\_x}}$ and $\boldsymbol{f}_{\mathrm{down\_v}}$ through ReLU layer as Eq.4.11. At last, $\boldsymbol{f}_{\mathrm{x\_v}}$ and $\boldsymbol{f}_{\mathrm{v\_x}}$ are respectively upsampled into D_{hidden} dimension and summed with the original output language feature $\hat{\boldsymbol{f}}^l_x$ and visual feature $\hat{\boldsymbol{f}}^l_v$ as the final output feature $\boldsymbol{f}^l_x$ and $\boldsymbol{f}^l_v$ of the l-th transformer block:

$$\hat{\boldsymbol{f}}^l_x, \hat{\boldsymbol{f}}^l_v = \mathrm{TRM}(\boldsymbol{f}^{l-1}_x, \boldsymbol{f}^{l-1}_v), \tag{4.22}$$

$$\boldsymbol{f}^l_x = \mathrm{LN}(\hat{\boldsymbol{f}}^l_x + \boldsymbol{W}^{\intercal}_{\mathrm{up\_x}} \boldsymbol{f}_{\mathrm{x\_v}}), \tag{4.23}$$

$$\boldsymbol{f}^l_v = \mathrm{LN}(\hat{\boldsymbol{f}}^l_v + \boldsymbol{W}^{\intercal}_{\mathrm{up\_v}} \boldsymbol{f}_{\mathrm{v\_x}}). \tag{4.24}$$

where $\mathrm{TRM}(\cdot)$ represents the transformer block in the cross-modal encoder.

Incorporating LoRA in VLN-PETL

As shown in Figure 4.4, VLN-PETL incorporates LoRA as an independent PETL component in addition to LEA, HIB and CIB to further improve the performance on downstream VLN tasks. Specifically, we globally inject LoRA layers into query matrices $\boldsymbol{W}_Q$ and value matrices $\boldsymbol{W}_V$ of all multi-head attention layers in the pre-trained model. Given the input feature $\boldsymbol{f}_{\mathrm{in\_q}}$ and $\boldsymbol{f}_{\mathrm{in\_v}}$ for linear projection in the multi-head attention layer, the output feature $\boldsymbol{Q}$ and $\boldsymbol{V}$ can be computed as follow:

$$\boldsymbol{Q} = (\boldsymbol{W}^{\intercal}_Q + \gamma \boldsymbol{W}^{\intercal}_{\mathrm{up\_q}} \boldsymbol{W}^{\intercal}_{\mathrm{down\_q}}) \boldsymbol{f}_{\mathrm{in\_q}}, \tag{4.25}$$

$$\boldsymbol{V} = (\boldsymbol{W}^{\intercal}_V + \gamma \boldsymbol{W}^{\intercal}_{\mathrm{up\_v}} \boldsymbol{W}^{\intercal}_{\mathrm{down\_v}}) \boldsymbol{f}_{\mathrm{in\_v}}. \tag{4.26}$$

Details of Language Encoder Adapter

Concretely, for the l-th transformer block in the language encoder, the input feature $\boldsymbol{f}^{l-1}_x$ is first passed through an adapter with D_{mid} bottleneck dimension to generate a new feature $\boldsymbol{f}_{\mathrm{ad\_att}}$. Then, $\boldsymbol{f}_{\mathrm{ad\_att}}$ is summed with the original output feature $\hat{\boldsymbol{f}}_{\mathrm{att}}$ of the multi-head self-attention layer in the transformer block as:

$$\boldsymbol{f}_{\mathrm{ad\_att}} = \boldsymbol{W}^{\intercal}_{\mathrm{up\_att}} \mathrm{ReLU}(\boldsymbol{W}^{\intercal}_{\mathrm{down\_att}} \boldsymbol{f}^{l-1}_x), \tag{4.27}$$

$$\hat{\boldsymbol{f}}_{\mathrm{att}} = \mathrm{MSA}(\boldsymbol{f}^{l-1}_x), \tag{4.28}$$

$$\boldsymbol{f}_{\mathrm{att}} = \mathrm{LN}(\hat{\boldsymbol{f}}_{\mathrm{att}} + \boldsymbol{f}_{\mathrm{ad\_att}}), \tag{4.29}$$

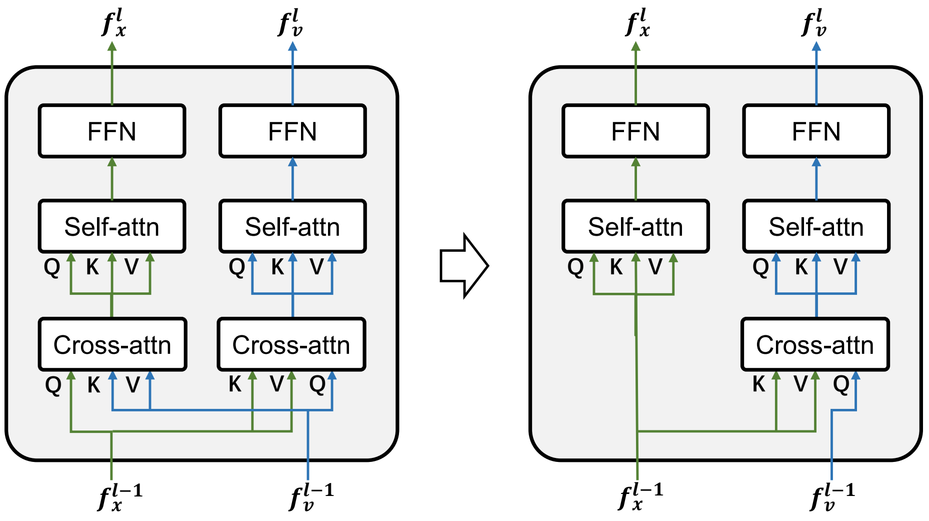


FIGURE 4.5. Variant of VLN baseline for RxR.

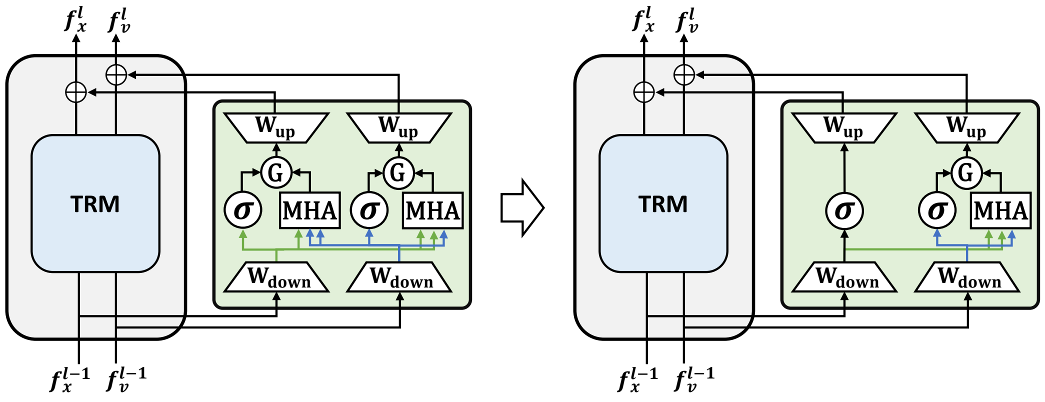


FIGURE 4.6. Variant of Cross-modal Interaction Booster for RxR.

where $\boldsymbol{W}_{\text{up\_att}}$ and $\boldsymbol{W}_{\text{down\_att}}$ represent projection matrices of the adapter, MSA$(\cdot)$ represents multi-head self-attention layer and LN$(\cdot)$ represents layer normalization. Note that we omit the feed-forward layer following multi-head self-attention layer for brevity.

Similarly, another adapter is inserted into the feed-forward layer which takes $\boldsymbol{f}_{\text{att}}$ as input:

$$\boldsymbol{f}_{\text{ad\_ffn}} = \boldsymbol{W}_{\text{up\_ffn}}^{\top}\text{ReLU}(\boldsymbol{W}_{\text{down\_ffn}}^{\top}\boldsymbol{f}_{\text{att}}), \tag{4.30}$$

$$\hat{\boldsymbol{f}}_{\text{ffn}} = \text{FFN}(\boldsymbol{f}_{\text{att}}), \tag{4.31}$$

$$\boldsymbol{f}_{\text{ffn}} = \text{LN}(\hat{\boldsymbol{f}}_{\text{ffn}} + \boldsymbol{f}_{\text{ad\_ffn}}), \tag{4.32}$$

where FFN(·) represents feed-forward layer. Finally, $\boldsymbol{f}_{\text{ffn}}$ is used as the output feature $\boldsymbol{f}_x^l$ of the l-th transformer block:

$$\boldsymbol{f}_x^l = \boldsymbol{f}_{\text{ffn}}. \tag{4.33}$$

Variant of VLN-PETL for RxR

To improve the efficiency of HAMT for VLN tasks with much longer instructions such as RxR, [15] adopts a structure variant for cross-modal encoder as shown in Figure 4.5. We follow this practice for our VLN baseline in the RxR task and correspondingly our proposed cross-modal Interaction Booster (CIB) also has a modification for the language sub-branch as shown in Figure 4.6. Specifically, for the language input feature $\boldsymbol{f}_x^{l-1}$, only a bottleneck layer with an activation layer works as a block-level adapter, excluding the multi-head cross-attention mechanism and gating mechanism. The forward pass of this variant can be formulated as follows:

$$\boldsymbol{f}_x^l = \text{LN}\big(\text{TRM}(\boldsymbol{f}_x^{l-1}) + \text{ADAPTER}(\boldsymbol{f}_x^{l-1})\big). \tag{4.34}$$

4.4 Experiments

4.4.1 Downstream tasks

To comprehensively evaluate PETL methods for VLN, we conduct experiments on four downstream tasks: Room-to-Room (R2R) [5], REVERIE [95], NDH [117] and Room-across-Room (RxR) [54]. These downstream tasks evaluate the agent from different views: (1) R2R and RxR require the agents to follow detailed instructions to navigate from one room to another; (2) REVERIE gives a concise, high-level instruction referring to a remote object, which focuses on grounding remote target objects; (3) NDH requires an agent to reach target regions based on the dialog history, which contains multiple question-and-answer interactions between the agent and an oracle.

4.4.2 Evaluation metrics

We follow previous work and adopt the most commonly used metrics for evaluating VLN agents as follows:

`TL` (Trajectory Length) measures the average length of all the predicted navigation trajectories in meters. `NE` (Navigation Error) is the mean the average distance in meters between the agent's final location and the target location. `SR` (Success Rate) measures the ratio of successful tasks, of which the agent's stop location is less than 3 meters away from the target location. `SPL` (Success weighted by Path Length [3]) trades-off SR (Success Rate) against TL (Trajectory Length), which measures both the accuracy and efficiency of navigation. `OSR` (Oracle Success Rate) measures the ratio of tasks of which one of its trajectory viewpoints can observe the target object

Table 4.1. Performance of PETL methods on R2R

Methods	Updated Params(%)	Validation Seen				Validation Unseen				Test Unseen			
		TL	NE ↓	SR ↑	SPL ↑	TL	NE ↓	SR ↑	SPL↑	TL	NE ↓	SR ↑	SPL ↑
Fine-Tuning	100	11.48	2.94	72.67	69.17	11.62	3.64	64.24	59.25	12.20	4.09	63.20	58.55
BitFit [7]	0.46	11.61	3.78	63.47	60.35	12.22	4.18	59.17	54.67	12.96	4.63	57.15	53.03
Prompt Tuning [58]	0.37	10.67	4.24	61.02	58.59	11.14	4.63	56.49	52.32	11.60	4.88	54.47	50.91
LoRA [41]	3.02	11.73	3.14	70.13	66.00	12.25	3.84	63.60	57.59	12.99	4.15	61.44	55.96
Adapter [40]	3.08	11.70	3.34	67.38	64.42	12.66	4.00	63.01	57.42	13.19	4.27	60.69	55.88
VLN-PETL(ours)	2.82	11.39	**2.93**	**72.28**	**68.50**	11.52	**3.53**	**65.47**	**60.01**	12.30	**4.10**	**63.22**	**58.25**

Table 4.2. Performance of PETL methods on REVERIE. SPL is the main metric for its navigation task, and RGSPL is the main metric for the object grounding task.

Methods	Updated	REVERIE Validation Seen						REVERIE Validation Unseen						REVERIE Test Unseen					
		Navigation				RGS↑	RGSPL↑	Navigation				RGS↑	RGSPL↑	Navigation				RGS↑	RGSPL↑
	Params(%)	SR↑	OSR↑	SPL↑	TL			SR↑	OSR↑	SPL↑	TL			SR↑	OSR↑	SPL↑	TL		
Fine-Tuning	100	46.73	52.35	42.75	13.37	30.64	27.91	32.63	37.82	28.92	15.66	18.66	16.06	33.09	37.82	27.02	12.83	15.04	13.32
BitFit [7]	0.80	33.52	37.81	31.84	11.45	19.96	18.93	24.65	27.46	21.34	12.36	10.85	9.43	21.50	24.95	18.85	12.53	9.87	8.62
Prompt Tuning [58]	0.71	25.86	33.17	23.21	12.15	8.29	7.44	19.94	25.08	17.75	12.43	5.82	5.07	19.95	24.24	17.94	11.61	5.51	4.88
LoRA [41]	3.33	42.30	46.24	38.63	12.89	29.87	27.45	29.42	34.28	26.17	15.96	15.25	13.45	**32.12**	37.00	**26.86**	14.93	14.94	12.76
Adapter [40]	3.39	40.76	45.05	37.43	13.75	27.13	24.62	29.48	32.83	26.62	14.59	16.05	14.21	29.20	32.31	24.78	14.96	14.51	12.48
VLN-PETL(ours)	2.81	**45.96**	51.23	**42.60**	12.86	**29.94**	**27.61**	**31.81**	37.03	**27.67**	14.47	**18.26**	**15.96**	30.83	36.06	26.73	14.00	**15.13**	**13.03**

within 3 meters. RGS (Remote Grounding Success rate) measures the ratio of tasks that successfully locate the target object. RGSPL (RGS weighted by Path Length) is RGS weighted by Path Length. GP (Goal Progress) measures the average progress of the agent towards the target. nDTW (Normalized Dynamic Time Warping) penalizes deviations from the reference path. sDTW (Success weighted by normalized Dynamic TimeWarping) constrains nDTW to only successful episodes and effectively captures both success and fidelity.

4.4.3 Implementation details

We choose existing PETL methods of BitFit, Prompt Tuning, Adapter and LoRA for comparison with our integrated VLN-PETL. We use the same learning rate of $1e-4$, and AdamW [73] optimizer for all PETL methods. The batch size is set as 4 for REVERIE and 8 for the other three VLN tasks. For Prompt Tuning, we respectively add 20 prompt tokens in front of the inputs of the language encoder and vision encoder. For the setting of both Adapter and LoRA, the bottleneck dimension D_{dim} is set as 64, which brings comparative parameters for a fair comparison with VLN-PETL. While for VLN-PETL, the bottleneck dimensions D_{dim} of the Language Encoder Adapter (LEA), History Interaction Booster (HIB), and Cross-modal Interaction Booster (CIB) is set as 64 while the bottleneck dimension of the incorporated LoRA is set as 8 for REVERIE while 16 for other VLN tasks. The attention heads number of both HIB and CIB is set as 4. For all PETL methods, the prediction heads of the pre-trained model are also trained.

TABLE 4.3. Performance on NDH measured by Goal Progress (m).

Methods	Updated Params(%)	Val Seen	Val Unseen	Test Unseen
Fine-Tuning	100	7.69	5.16	5.05
BitFit [7]	0.46	<u>6.68</u>	3.77	4.03
Prompt Tuning [58]	0.37	4.71	3.26	2.86
LoRA [41]	3.02	6.83	5.16	<u>5.91</u>
Adapter [40]	3.08	5.29	<u>5.30</u>	4.72
VLN-PETL(ours)	2.82	**7.76**	**5.69**	**6.13**

4.5 Experimental Results

4.5.1 Comparison of PETL methods for VLN

As shown in Table 4.1-4.4, we compare our proposed VLN-PETL with finetuning, Bitfit [7], Prompt Tuning [58], LoRA [41], and Adapter [40] in performance and trainable parameter amounts on different VLN tasks.

We can see that Prompt Tuning with the least updated parameters works poorly for all VLN downstream tasks. One possible reason may be the limited trainable parameters. More importantly, the training instability of Prompt Tuning as previous work declared should also be responsible for the poor performance. In fact, training VLN agents itself is not always stable and easy, where reinforcement learning plays a vital role. BitFit surpasses Prompt Tuning with a non-trivial margin on all tasks with comparable amounts of trainable parameters. However, the performance of BitFit still falls far from finetuning, LoRA and Adapter on all VLN tasks especially on RxR in high demand for language understanding with much longer instructions. Only tuning bias terms may have difficulties in handling these complex VLN tasks. Thus, we believe that Prompt Tuning and BitFit are not applicable for efficiently tuning large pre-trained models for challenging VLN tasks. While LoRA and Adapter not only have comparative amounts of trainable parameters but also have comparable performances on all VLN tasks. These two methods further shrink the performance gap with finetuning, which are potential to effectively tune VLN pretrained models.

As for VLN-PETL, though it has fewer parameters than LoRA and Adapter, VLN-PETL still surpasses LoRA and Adapter on most evaluation metrics in all four downstream VLN tasks. Furthermore, only VLN-PETL maintains competitive performances compared to fine-tuning and even outperforms fine-tuning on several evaluation metrics. As shown in Table 4.3, it is worth mentioning that VLN-PETL outperforms full fine-tuning on all dataset splits in the NDH task, and achieves the top position on the public leaderboard. These promising results demonstrate the effectiveness of our proposed VLN-PETL for efficiently tuning large pre-trained models for VLN tasks.

TABLE 4.4. Performance on RxR using English instructions. nDTW is the main metric for the RxR task.

Methods	Updated Params(%)	RxR Validation Seen				RxR Validation Unseen			
		SR↑	SPL ↑	nDTW ↑	sDTW ↑	SR↑	SPL ↑	nDTW ↑	sDTW↑
Fine-Tuning	100	64.93	61.28	69.26	55.92	57.88	54.18	64.52	49.44
BitFit [7]	0.28	35.69	33.58	49.37	29.81	36.63	34.34	50.47	30.49
Prompt Tuning [58]	0.23	27.92	25.98	42.93	22.97	29.95	27.72	44.94	24.67
LoRA [41]	1.86	55.66	52.41	63.20	47.31	54.53	51.14	63.23	46.94
Adapter [40]	1.90	58.52	54.96	65.14	50.23	55.19	51.44	63.56	47.32
VLN-PETL(ours)	1.67	**60.48**	**56.77**	**65.74**	**51.67**	**57.95**	**54.16**	**64.94**	**49.70**

TABLE 4.5. Ablation of different components in VLN-PETL on REVERIE Unseen set and RxR Unseen set.

Components			REVERIE Val Unseen		RxR Val Unseen	
LEA	HIB	CIB	SPL↑	RGSPL↑	SR↑	SPL↑
×	×	×	15.19	4.53	28.66	26.72
✓	×	×	18.21	9.73	43.45	40.68
×	✓	×	18.49	9.32	41.04	38.46
×	×	✓	21.62	12.86	52.01	48.82
✓	✓	×	21.86	11.75	45.08	42.11
✓	×	✓	25.00	14.34	52.26	49.26
×	✓	✓	25.55	14.89	54.26	50.71
✓	✓	✓	26.51	15.29	56.04	52.79

TABLE 4.6. Performance comparison of HIB and CIB with their counterparts of Adapter on REVERIE val set.

Methods	Validation Seen		Validation Unseen	
	SPL↑	RGSPL↑	SPL↑	RGSPL↑
HEA	25.74	12.51	18.98	8.39
HIB	28.55	16.95	18.49	9.32
CEA	31.44	17.55	20.77	10.14
CIB	38.99	25.46	21.62	12.86

4.5.2 Ablation Study

Contribution of VLN-PETL components

As shown in Table 4.5, to evaluate the contribution of LEA, HIB and CIB, we choose REVERIE and RxR which are more challenging VLN tasks to conduct ablation studies. REVERIE not only measures the agent's ability in navigation but also in locating the target object, while RxR has much longer instructions requiring comprehensive language understanding. We also report the results of only tuning the prediction head for comparison. We find that LEA has a competitive performance compared to HIB, and only tuning either LEA or HIB outperforms the head tuning with a nontrivial margin. While CIB contributes much more inefficiently tuning the VLN model, which improves the performance with a larger margin on both REVERIE and RxR. This result indicates that language understanding and vision understanding with history

TABLE 4.7. Ablation of LoRA's effect on REVERIE val set.

Methods	Validation Seen		Validation Unseen	
	SPL↑	RGSPL↑	SPL↑	RGSPL↑
VLN-PETL	42.60	27.61	27.67	15.96
w/o LoRA	41.34	27.12	26.51	15.29

TABLE 4.8. Ablation of T and α in the gates of HIB and CIB on REVERIE Val Unseen set.

T	0.01	0.1	1	10
SPL↑	25.60	27.67	**27.81**	25.00
RGSPL↑	14.41	**15.96**	15.36	14.06

α	0.5	learnable
SPL↑	27.51	**27.67**
RGSPL↑	14.50	**15.96**

knowledge contribute comparably to the action prediction for VLN agents, while the cross-modal interaction plays more importantly in this process. By combining all these three components, the VLN agent achieves a promising performance and outperforms other PETL methods, especially on the RGSPL metric, which measures the agent's ability to locate the target object.

Superiority of HIB and CIB

As shown in Table 4.6, to validate the effectiveness of HIB and CIB, we compare the performance of HIB and CIB with their counterparts of Adapter, by respectively replacing HIB and CIB by History Encoder Adapter (HEA) and Cross-modal Encoder Adapter (CEA) which are similar to Language Encoder Adapter. Due to the enhancement of historical knowledge learning, HIB surpasses HEA on seen set by a large margin. On the unseen set, HIB falls behind HEA with a trivial margin on the SPL metric while outperforming HEA on the RGSPL metric with a large margin. This is probably because the input for history encoding is a panoramic view image rather than a single-view image of the front view, where HIB tends to learn more knowledge about the fine-grained object rather than the trajectory. As for CIB and CEA, CIB surpasses CEA on all metrics and all sets by a large margin, which shows the superiority of CIB.

The Effect of LoRA

As shown in Table 4.7, we find that when removing LoRA, the performance of VLN-PETL has a slight drop on all main metrics on both REVERIE seen and unseen splits. Besides, the decrease on RGSPL metric is less than that on SPL metric, which indicates LoRA's effect on the VLN agent's ability to locate objects is smaller than that of navigation during efficient tuning.

Hyper-parameters in Gates.

As shown in Table 4.8, we conduct ablation studies on T and α in the gates of HIB and CIB. The performances are similarly high when T is set as 0.1 or 1. We set T as 0.1 due to its higher score on RGSPL. We also compare the results of fixing α as 0.5 and using the learnable gate. We can see that the learnable gate α surpasses the fixed α with a large margin on RGSPL.

4.6 Conclusion

In this paper, we present the first study of applying Parameter-Efficient Transfer Learning (PETL) methods to VLN tasks and propose a VLN-specific PETL method named VLN-PETL. Considering the characteristics of VLN, we specifically design two PETL modules to efficiently tune the large pre-trained model for VLN downstream tasks, namely Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). Both HIB and HIB mainly consist of bottleneck layers and multi-head attention layers, which respectively enhance the vision encoder's learning of history knowledge and the cross-modal encoder's interactions between the language and vision features during the efficient tuning. In addition, we incorporate the vanilla adapters to efficiently tune the language encoder and the LoRA to further improve the integrated performance. Extensive experiments conducted on four mainstream VLN tasks of R2R, REVERIE, NDH, and RxR show the effectiveness of our proposed VLN-PETL. Furthermore, we conduct ablation studies to evaluate the contribution of VLN-PETL components and validate the superiority of our specifically designed HIB and CIB to their counterpart PETL methods.

Statement of Authorship

Title of Paper	VLN-PETL: Parameter-Efficient Transfer Learning for Vision-and-Language Navigation		
Publication Status	☐ Published	☐ Accepted for Publication	
	[✓ Submitted for Publication]	☐ Unpublished and Unsubmitted work written in manuscript style	
Publication Details	Submitted to ICCV 2023.		

Principal Author

Name of Principal Author (Candidate)	Yanyuan Qiao		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	28/06/2023

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Zheng Yu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Qi Wu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	29/06/2023

Please cut and paste additional co-author panels here as required.

Chapter 5

Interactive Prompting for Remote Embodied Referring Expression

In addition to the VLN pre-trained model in Chapter 3 and Parameter-Efficient Transfer Learning (PETL) techniques for VLN in Chapter 4, another challenge remains in effectively leveraging the knowledge embedded within Large Language Models (LLMs) to enhance action prediction in VLN tasks. To tackle this challenge, in this chapter, we introduce a model named March-in-Chat. This model enables the agent to engage in interactive conversations with an LLM, facilitating the generation of fine-grained plans that guide the agent's navigation process.

5.1 Introduction

Vision-and-Language Navigation (VLN), which lies at the intersection of computer vision, natural language processing, and robotics, has aroused great attention from research communities in the past few years. Given instructions in natural language, the VLN agent should navigate to the target location based on the dynamic observations in the 3D simulated environments. Since VLN has great potential in real-world applications such as domestic assistant robots, a large amount of specific VLN tasks have been proposed, including R2R [5] and RxR [54] that ask the agent to navigate from one room to another in a photo-realistic environment according to the fine-grained instruction, NDH [117] provides detailed dialogues which imply the instruction, TouchDown [12] extends the task into an outdoor environment, REVERIE [95] and SOON [135] that additionally require the agent's ability of remote object grounding and ALFRED [107] that asks the agent to interact with the target object in a single room of the synthetic environment.

Most of these VLN tasks provide detailed step-by-step instructions to the agent, such as "Go up the stairs and then walk the length of the couch. Walk past the dining area and into the kitchen. Stop in front of the refrigerator." in R2R. Although detailed instructions can help the agent better achieve the navigation goal in the simulated environments, it has a big gap towards real applications where human beings tend

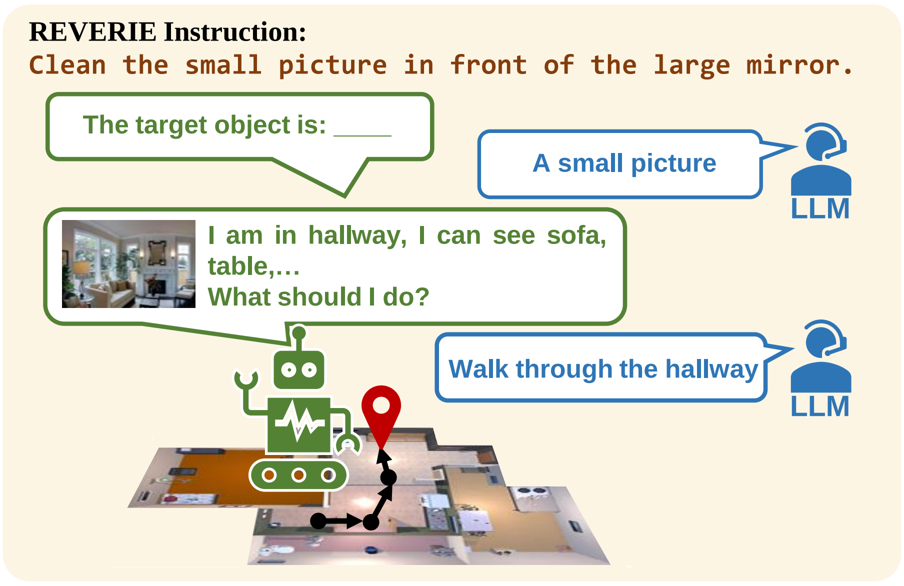

REVERIE Instruction:
Clean the small picture in front of the large mirror.
The target object is: ____
A small picture
LLM
I am in hallway, I can see sofa, table,…
What should I do?
Walk through the hallway
LLM

FIGURE 5.1. Our March-in-Chat (MiC) model is talking to a Large Language Model (LLM) to generate navigation plans on the fly, with the REVERIE instruction and the dynamic room-and-object information as inputs.

to give coarse-grained high-level instructions such as "Go to the refrigerator on the second floor". Contrary to other tasks, the Remote Embodied Referring Expression (REVERIE) task is more likely to empower the real-world applications of VLN, of which the instructions are closer to those in practice, such as "Empty the washing machine on level one". Such high-level instruction is more challenging for VLN agents since it requires them to be more competent in perceiving the surrounding environment and the navigation progress and correspondingly making reasonable plans for the next steps.

Recently, Large Language Models (LLMs) that internalize a wealth of commonsense knowledge show great potential in action planning for some embodied tasks with the help of suitable in-context learning. However, previous works mainly utilize LLMs to plan atomic actions of object manipulation in a very limited space with simple scenes. These predefined atomic actions can be easily planned well by the LLMs planners with a unified template. Different from these embodied tasks, REVERIE requires large-area exploration from one room to another, which is complex in the layout of rooms and scenes with diverse objects.

In this work, to adapt LLMs as the planner for REVERIE with the ability of comprehensive scene perception, we propose a novel model named *March in Chat* (MiC), which enables the LLM as an environment-aware instruction planner through on-the-fly dialogues between the agent and the LLM as Fig. 5.1 shows. Specifically, the agent is initially situated at the starting position given a high-level coarse-grained REVERIE instruction. First, a Goal-Oriented Static Planning (**GOSiP**) module queries the LLM to point out the target object and reason out where the thing may be by arousing the rich world knowledge internalized in the LLM. Secondly, the agent's Room-and-Object Aware Scene Perceiver (**ROASeP**) describes the current observation and asks the LLM to generate step-by-step fine-grained planning for the next navigation steps. Then, if the ROASeP finds the room has changed, the LLM is queried again by the Scene-Oriented Dynamic Planning (**SODiP**) module to generate a new fine-grained step-by-step planning, which will be concatenated with all previous responses from the LLM. The agent will march through such multi-round dialogues until the task is finished, under the guidance of interactive prompting.

To evaluate our proposed MiC, we conduct experiments on the REVERIE benchmark. Our MiC achieves a new state-of-the-art performance in all metrics on REVERIE val unseen set and REVERIE test unseen set. Mainly, MiC obtains 41.97% on the primary navigation metric of SPL and 26.17% on the major object grounding metric of RGSPL on test split, which is at least 3.09% and 3.49% higher than the previous SoTA results. We also conduct ablation studies to validate the contributions of different components in MiC and the effect of scene-aware perception in dynamic planning generation. These promising results demonstrate the effectiveness of our proposed MiC.

In summary, we make the following contributions:

- We propose a novel March-in-Chat (MiC) model, which lets the REVERIE agent talk with an LLM on the fly to make plans for the next few steps.
- Two planning modules, namely Goal-Oriented Static Planning (GOSiP) module, and Scene-Oriented Dynamic Planning (SODiP) module, and one Room-and-Object Aware Scene Perceiver (ROASeP) module, are proposed.
- Extensive quantitative and qualitative experiments are conducted on REVERIE to validate the effectiveness of our method.

5.2 Background

5.2.1 Remote Embodied Reffering Expression

In the REVERIE task, given a concise and high-level instruction referring to a remote object, the agent is expected to navigate to the goal location and identify the target object in previously unseen environments. The environment is defined as an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{V_i\}_{i=1}^{K}$ denotes K navigable nodes, and $\mathcal{E}$ denotes connectivity edges. The agent is first placed in a starting node with the initial state s_0 and perceives a panorama $\mathcal{R}_t$ as the visual observation at each time step t. The panorama $\mathcal{R}_t$ is split into n single view images as $\mathcal{R}_t = \{r_i\}_{i=1}^{n}$. Each single view image r_i is represented by an image feature vector and an orientation feature vector. In addition, the object features $\mathcal{O}_t = \{o_i\}_{i=1}^{m}$ of m objects are extracted from the panorama view using the annotated object bounding boxes or object detectors. Then, the agent makes a sequence of actions $\langle a_0, ...a_T \rangle$ to reach the target location, where each action is achieved by choosing a navigable node from the candidate list. The agent navigates in the environment until the target object is grounded or the agent reaches the pre-defined maximum trajectory length.

5.2.2 Agent

HM3D-DUET is a dual-scale graph transformer with topological maps, which contains two modules: topological mapping and global action planning. The topological mapping module builds a topological map during navigation. And the global action planning module predicts the next location on the map or a stop action to end the navigation.

Topological Mapping

To build the environment graph $\mathcal{G}$ which is unknown initially, the mapping module updates node representations by adding the newly observed location gradually to the map. Specifically, the map denote as $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$. At time step t, the current node V_t and its neighboring unvisited nodes $\mathcal{N}(V_t)$ are added to $\mathcal{V}_{t-1}$.

The mapping module outputs the current panorama encoding with image features $\{r_i\}_{i=1}^{n}$ and object features $\{o_i\}_{i=1}^{m}$ and a graph with K node features $\{v_i\}_{i=1}^{K}$.

Global Action Planning

The module uses dual-scale architecture transformers to capture cross-modal vision-and-language relations from different scales: a fine-scale representation of the current location and a coarse-scale representation of the map.

In the coarse-scale cross-modal encoder, the inputs are map node features $\{v_i\}_{i=1}^{K}$ and textual features $\mathcal{T}$. The node features are embedded and combined with word embeddings into a multi-layer graph-aware cross-modal transformer to get node embedding $\widehat{v_i}$. Then the node embedding $\widehat{v_i}$ is fed into a two-layer feed-forward network (FFN) to predict a navigation score for each node s_i^c.

In the fine-scale cross-modal encoder, the inputs are fine-grained visual representations $\{\mathcal{R}_t, \mathcal{O}_t\}$, the textual features $\mathcal{T}$, and a special stop token r_0. Then the concatenated visual tokens $[r_0; \mathcal{R}_t; \mathcal{O}_t]$ and textual features $\mathcal{T}$ are fed into a standard multi-layer cross-modal transformer to get $[\widehat{r_0}; \widehat{R_t}; \widehat{O_t}]$. The navigation score for local-level s_i^f and object are predicted via FFN, a similar way in the coarse-scale cross-modal encoder.

5.2.3 LLMs as Embodied Planner

Benefiting from the rise of LLMs, recent works [43, 108] have explored the use of LLMs in task planning for various embodied tasks. Huang *et al.* [43] propose to utilize the frozen LLMs (*e.g.*, GPT-2 [98], GPT-3 [9] and Codex [14]) to plan actions for the embodied agent with in-context learning [9]. SayCan [1] translates a high-level instruction into a list of candidate low-level actions with a probability, which is then multiplied by a value function for action prediction. These two LLM planners are static, which only generate action plans at the beginning of a task. By contrast, Huang *et al.* [44] propose to introduce the feedback of action progress, detected objects and human assistance into the LLM planner to re-plan atomic actions. One concurrent work by Song *et al.* [109] injects the detected objects to re-generate high-level plans with a fixed program pattern for the ALFRED [107] task.

However, these above-mentioned methods mainly concentrate on planning atomic actions for object manipulation in a very limited space with simple scenes. By contrast, REVERIE has plenty of much larger and more complicated environments: 90 multi-layer buildings of various styles (*e.g.*, office, home, gym, to name a few). To handle the complex scenarios of REVERIE, we propose a Room-and-Object Aware Scene Perceiver module that helps the LLM planner dynamically interact with the environment in the form of a natural language dialogue.

5.3 Method

As illustrated in Fig. 5.2, when initially situated at the starting position and given a concise high-level instruction such as "Empty the washing machine on level one", the agent first queries an LLM with the **GOSiP** module (Sec. 5.3.1) to find out the target object "`washing machine`" in the instruction and reason out the potential location "`laundry room`" by arousing the world knowledge implied in an LLM. Then, the **ROASeP** module (Sec. 5.3.2) obtains the environmental feedback, which extracts the room type and visible objects from the current visual observation. With the description of the scene perception, the LLM is queried again by the **SODiP** (Sec. 5.3.1) to generate the next step instruction, which is used to guide the agent to navigate to the target object in the room. The process of MiC is described in Algorithm 1.

5.3.1 Planning with World Knowledge from LLMs

This section illustrates how we utilize in-context learning (ICL) [9] to acquire world knowledge from the LLMs for planning. We first briefly introduce in-context learning. Then, we elaborate the Goal-Oriented Static Planning (GOSiP) and Scene-Oriented Dynamic Planning (SODiP). Last, we show the demonstration selection process.

Preliminary: In-context Learning for Planning.

In-context learning (ICL) is a paradigm that lets LLMs directly make predictions based on a natural language context without gradient updates [9].

Specifically, under the setting of in-context learning, an LLM is fed a "`prompt`" that usually contains a task description and several demonstrations, and then the

Algorithm 1 March in Chat

```
Notation Summary:
$I$: high-level instruction in REVERIE
$\mathcal{R}_t$: visual observation at timestep $t$
$\mathcal{D}$: demonstration set
$P_\text{o}$: prompt for LLM to generate planning
LLM: large language model
Template: templates to generate natural language description

$t \leftarrow 0$                                                      ▷ Initial timestep
$W_I \leftarrow I$
$\hat{o} \leftarrow \text{LLM}(I, P_\text{o})$                       ▷ Target object recognition
$\hat{l} \leftarrow \text{LLM}(\hat{o}, P_\text{l})$                 ▷ Target location reasoning
$W_G \leftarrow \text{Template}(\hat{o}, \hat{l})$                   ▷ GOSiP
$W_S \leftarrow \phi$
$W \leftarrow \text{Concat}(W_I, W_G, W_S)$                          ▷ Assembled instruction
$P_\text{demon} \leftarrow \text{DynamicSelect}(I, \mathcal{D})$     ▷ Dynamic demonstration
while $t <$max-step and $\hat{a}_t \neq$ "stop" do
    $\hat{c}^t_\text{room}, \hat{c}^t_\text{obj} \leftarrow \text{CLIP}(\mathcal{R}_t)$   ▷ ROASeP
    if $\hat{c}^t_\text{room} \neq \hat{c}^{t-1}_\text{room}$ then
        $P_\text{scene} \leftarrow \text{Template}(\hat{c}^t_\text{room}, \hat{c}^t_\text{obj})$
        $P_\text{step} \leftarrow \text{Template}(I, W_S)$
        $P_\text{SODiP} \leftarrow \text{Concat}(P_\text{scene}, P_\text{demon}, P_\text{step})$
        $I_\text{step} \leftarrow \text{LLM}(P_\text{SODiP})$           ▷ SODiP
        $W_S.\text{Append}(I_\text{step})$
        $W.\text{Update}(W_S)$                                        ▷ Instruction update
    end if
    $\hat{a}_t \leftarrow \text{Agent}(W, \mathcal{R}_t)$
    $t \leftarrow t + 1$
end while
```

LLM generates the required outputs. Both the prompt template and the choice of demonstration examples have an impact on how well ICL performs. In this work, we use two different ICL settings to generate different navigation plans, *i.e.* the GOSiP and SODiP module.

The GOSiP aims to identify the target object and reason out the target location by arousing the world knowledge implied in an LLM through appropriate prompts. A fixed demonstration example is used for GOSiP. While the SODiP aims to generate step-by-step planning instructions after observing the dynamic scenes from the environment, which is more complicated than the former. To better generate planning, we dynamically select the most suitable demonstration examples for SODiP and incorporate the environmental feedback as prompts for interactive planning.

Goal-Oriented Static Planning (GOSiP)

Given a high-level concise instruction, such as "Empty the washing machine on level one", an LLM is first asked to generate a goal-oriented static planning instruction: "Goal: The target object is a washing machine. It is usually in a laundry room", which emphasizes the target object and points out where the target object may lie. As

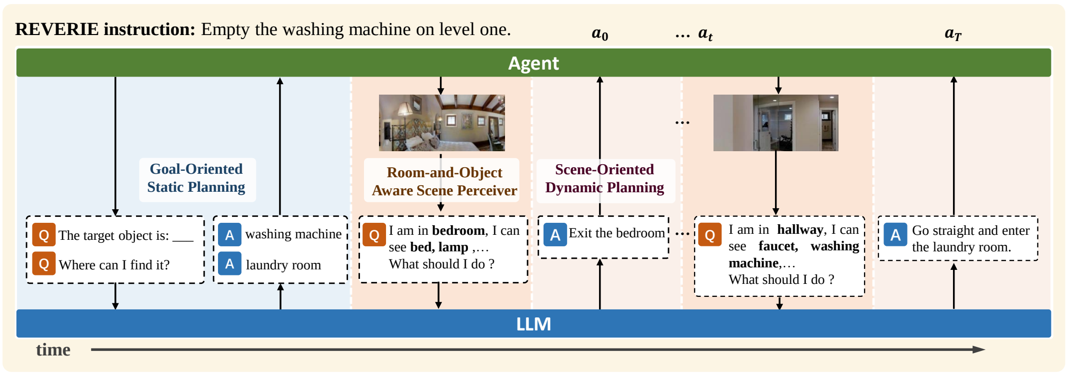
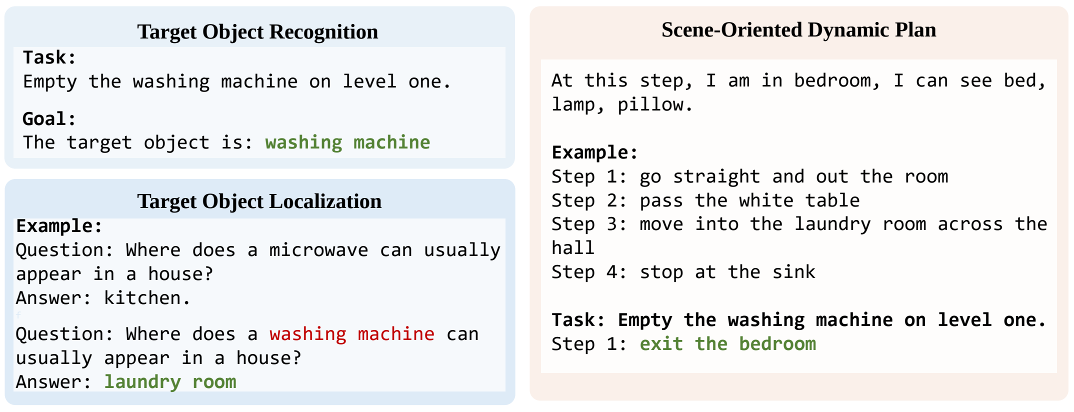

REVERIE instruction: Empty the washing machine on level one.
a_0
... a_t
a_T
Agent
Goal-Oriented Static Planning
Room-and-Object Aware Scene Perceiver
Scene-Oriented Dynamic Planning
Q The target object is: ___
Q Where can I find it?
A washing machine
A laundry room
Q I am in bedroom, I can see bed, lamp ,... What should I do ?
A Exit the bedroom
Q I am in hallway, I can see faucet, washing machine,... What should I do ?
A Go straight and enter the laundry room.
LLM
time

FIGURE 5.2. Overview of our March-in-Chat model. The program runs along the vertical arrows from left to right, progressing with the time flow. Our model first performs Goal-Oriented Static Planning (GOSiP, Sec. 5.3.1) to reason the target object and its possible lying room; then the Room-and-Object Aware Scene Perceiver (SOASeP, Sec. 5.3.2) perceives what room type the agent currently stands in and what prominent objects can be seen; these information are used by the Scene-Oriented Dynamic Planning module (SODiP, Sec. 5.3.1) to generate a detailed instruction to execute. GOSiP just runs once, and we repeat SOASeP and SODiP until the agent chooses to stop or reaches the maximum steps.

shown in Fig. 5.3(a), the planning generation mainly consists of two sub-tasks: target object recognition and target object localization, which can be achieved by providing specifically designed prompts for the LLM. To this end, we design the prompts for the former sub-task in the form: "**Task:** Empty the washing machine on level one. **Goal:** The target object is: ". Then the LLM will generate a corresponding answer: "`washing machine`". By contrast, the latter sub-task, reasoning out the target location, is more complex because it requires more suitable prompts to arouse the internalized world knowledge in the LLM. To address this problem, we utilize a fixed demonstration example for the LLM, and design the prompts for the latter sub-task in the form: "**Example:** Question: Where does a microwave can usually appear in a house? Answer: kitchen. Question: Where does a `washing machine` can usually appear in a house? Answer: ". Given such prompts, the LLM will generate the corresponding answer "`laundry room`". With these answers, we can easily combine them into the goal-oriented planning format: "Goal: The target object is a `washing machine`. It is usually in a `laundry room`".

Scene-Oriented Dynamic Planning (SODiP)

As is shown in Fig 5.3(b), the prompt for SODiP consists of three parts. The first part is based on the scene perception of room type, such as "`bedroom`", and visible objects, such as "`bed, lamp, pillow`", obtained by the Room-and-Object Aware Scene Perceiver (ROASeP, Sec. 5.3.2).



Target Object Recognition

```
Task:
Empty the washing machine on level one.

Goal:
The target object is: washing machine
```

Target Object Localization

```
Example:
Question: Where does a microwave can usually
appear in a house?
Answer: kitchen.

Question: Where does a washing machine can
usually appear in a house?
Answer: laundry room
```

Scene-Oriented Dynamic Plan

```
At this step, I am in bedroom, I can see bed,
lamp, pillow.

Example:
Step 1: go straight and out the room
Step 2: pass the white table
Step 3: move into the laundry room across the
hall
Step 4: stop at the sink

Task: Empty the washing machine on level one.
Step 1: exit the bedroom
```

(a) Prompt template of Goal-Oriented Static Planning **(b) Prompt template of Scene-Oriented Dynamic Planning**

FIGURE 5.3. Examples of prompting templates for Goal-Oriented Static Planning (a) and Scene-Oriented Dynamic Planning (b). Outputs are marked with green color. And the red denotes the predicted object from Target Object Recognition.

These information are transformed into a natural language description of the current scene in the format of "At this step, I am in `bedroom`, I can see `bed, lamp, pillow`". The second part is a demonstration of the fine-grained step-by-step instruction, which is selected according to the strategy detailed in the next section. The last part is previous instructions, such as "**Task**: Empty the washing machine on level one. Step 1: ". All these three parts are concatenated together and then fed into an LLM to generate the fine-grained planning instruction for the next step accordingly, such as "`Exit the bedroom`".

Dynamic Demonstration Selection

Recent works show that providing various demonstration examples to LLMs benefits the in-context learning for different tasks [43, 70, 91]. In light of these findings, to direct the LLMs in generating better fine-grained plannings, we dynamically select the most suitable demonstration example for each specific task in REVERIE as the prompt to generate the environment-aware instruction, contrary to using a single fixed demonstration for all tasks.

Specifically, we choose the training set of the Fine-Grained R2R dataset (FGR2R) [37] as the demonstration set $\mathcal{D}$, of which each sample will be used as a demonstration example D_{step}. As shown in Fig. 5.4, FGR2R decomposes each low-level instruction I_{low} of R2R dataset [5] into step-by-step instructions I_{step}. Then, given a high-level instruction I_{high} of REVERIE, a proper I_{low} will be selected as the demonstration example D_{step} by a matching algorithm. In particular, we use I_{high} as query Q and each low-level instruction I_{low} as the key K_i, both of which are embedded by the Sentence-BERT [101]. The semantic distance score between the two embeddings is

R2R
Go up the stairs and then walk the length of the couch. Walk past the dining area and into the kitchen. Stop in front of the refrigerator.

FGR2R
Step 1: go up the stair
Step 2: and then walk the length of the couch
Step 3: walk past the dining area and into the kitchen
Step 4: stop in front of the refrigerator

FIGURE 5.4. Example of instructions in R2R and FGR2R.

calculated by the cosine similarity:

$$\mathrm{s}(Q, K_i) = \frac{e(Q) \cdot e(K_i)}{\|e(Q)\| \|e(K_i)\|}, \tag{5.1}$$

where $e(\cdot)$ is the embedding function. If K_i has the highest similarity score to the given query Q, its corresponding step-by-step instruction I_{step} will be selected as the demonstration example D_{step} for the given high-level instruction I_{high}.

5.3.2 Room-and-Object Aware Scene Perceiver

Though the world knowledge acquired from the static LLMs planner could benefit the embodied task promisingly, the static LLMs planner may generate wrong or irrelevant plannings, which misleads the agent. To address this issue, the LLM planner should be aware of and interact with the dynamic observations. In [109], the names of objects obtained from the ground truth or pre-trained detectors have been added to in-context prompts. However, the agents of these works act in a very limited space with simple scenes and monotonous objects. By contrast, REVERIE involves large-area exploration between different floors and rooms, where the scenes are more complex with more diverse objects. Considering these factors, we propose a room-and-object aware scene perceiver (ROASeP) for the LLM planner, which predicts not only the room type but also the visible objects of the current location. Rather than using separate classifiers and detectors to individually predict each position's room types and visible object categories, we use CLIP [97] as the proposed room-and-object aware scene perceiver. Thanks to CLIP's strong ability of zero-shot image classification in the open world, the ROASeP can well handle these two tasks.

Specifically, we first fetch the room type labels from the MatterPort [11] semantic annotations and the object type labels are extracted from the REVERIE training dataset. They are used to build the codebook for the room categories $\mathbf{C}_{\text{room}}$ and the object categories $\mathbf{C}_{\text{obj}}$, respectively. Then, at each timestep t, the agent perceives the environment and obtains the panoramic visual observation $\mathcal{R}_t = \{r_i\}_{i=1}^{n}$. For each single-view observation r_i in the panorama, the image feature f_{r} is extracted by the CLIP Image Encoder

$$f_{\mathrm{r}} = E^{v}_{\mathrm{CLIP}}(r_i), \tag{5.2}$$

where $E^{v}_{\mathrm{CLIP}}(\cdot)$ represents the CLIP Image Encoder. For each room category c_{room} and each object category c_{obj}, we respectively construct a text phrase of room T_{room} as "a photo of a $\{c_{room}\}$" and a text phrase of object T_{obj} as "a photo of a $\{c_{obj}\}$". Then the text feature is derived through the pretrained CLIP Text Encoder as:

$$f_{\mathrm{room}} = E^{t}_{\mathrm{CLIP}}(T_{\mathrm{room}}), \tag{5.3}$$

$$f_{\mathrm{obj}} = E^{t}_{\mathrm{CLIP}}(T_{\mathrm{obj}}), \tag{5.4}$$

where $E^{t}_{\mathrm{CLIP}}(\cdot)$ represents the CLIP Text Encoder. At last, the similarity score S_{room} between the image feature f_{r} and the text feature f_{room} as well as the similarity score S_{obj} between the image feature f_{r} and the text feature f_{obj} are respectively computed as:

$$S_{\mathrm{room}} = \mathrm{Softmax}(f_{\mathrm{room}} \cdot f_{\mathrm{r}}^{T}), \tag{5.5}$$

$$S_{\mathrm{obj}} = \mathrm{Softmax}(f_{\mathrm{obj}} \cdot f_{\mathrm{r}}^{T}). \tag{5.6}$$

Considering that the current environment normally belongs to only one type of room, though the panoramic images have multiple views, the room that the agent is currently centered in should have the largest influence on each view. Thus, we average the predicted room type scores S_{room} from multiple views and choose the room type with the greatest score as the room type prediction $\hat{c}_{\mathrm{room}}$. For object predictions, if the object occupies more proportion in a view, the matching score S_{obj} should be higher. Thus, we select k prominent objects with the top-k matching scores as the auxiliary environment feedback in addition to the predicted room.

5.3.3 March with Interactive Prompting

When the generation of the goal-oriented planning and the scene-oriented planning with perceptions from the environment is finished, the agent can march towards the target object at each timestep t under the guidance of the interactive prompting. In this section, we will give a detailed description of how the interactive prompting works during the process of navigation, which mainly consists of two parts, *i.e.* the assembled instruction and the instruction update.

Assembled Instruction

At each timestep t, the agent observes the environment and receives the assembled instructions obtained from the above-mentioned modules, and chooses an action to

HLI	Empty the washing machine on level one.
GOSiP	Goal: The target object is washing machine. It is usually in laundry room.
SODiP	Step 1: exit the bedroom Step 2: go down the stairs …

Figure 5.5. Text inputs contains three parts: High-level Instruction in REVERIE (HLI), Goal-Oriented Static Planning (GOSiP) and Scene-Oriented Dynamic Planning (SODiP) returned instructions.

perform. Specifically, as shown in Fig. 5.5, the assembled instructions W of the interactive prompting mainly consist of three parts: the high-level instruction (HLI) W_I in REVERIE, the GOSiP instruction W_G and the SODiP instruction W_S. We concatenate these three parts of instructions as the assembled instruction $W = [W_I, W_G, W_S]$ and use WordPieces [47] to tokenize all the words into a sequence of tokens as the textual input for the agent. Then, the agent will act under the guidance of such assembled instruction. Note that the use of the original high-level instruction W_I can improve the model's tolerance on the noise of intermediate planning instructions.

Instruction Update

The GOSiP is only conducted once at the beginning of the task. While the SODiP is conducted depending on the feedback of environments. Specifically, at each timestep t, if the ROASeP finds the room has changed where the predicted room $\hat{c}^t_{\text{room}}$ does not equal to $\hat{c}^{t-1}_{\text{room}}$, the SODiP will be triggered again. Then, a new step-by-step instruction such as "Step 2: go down the stairs" for the next few steps will be generated by the LLM and added to the previous assembled instruction W after the last step-by-step instruction of "Step 1: exit the bedroom". Then, the agent will act under the guidance of the updated instructions W'.

5.4 Experiments

5.4.1 Evaluation Setup

Dataset

REVERIE [95] contains 10,567 panoramic images within 90 buildings (4,140 target objects divided into 489 categories) and 21,702 instructions with 18 words on average. Each target viewpoint has 7 distinct panoramic objects with 50 bounding boxes on average. It consists of four splits: train, validation seen, validation unseen and test unseen.

Evaluation Metrics

The performance of agents is evaluated in two ways: navigation and object grounding. For the navigation sub-task, the metrics are **Success Rate (SR)**, **Oracle Success Rate (OSR)**, and **Success weighted by Path Length [3] (SPL)**, where SPL is the main metric. For the grounding sub-task, the metrics are **Remote Grounding Success rate (RGS)** and **RGS weighted by Path Length (RGSPL)**, where RGSPL is the main metric for this sub-task. For all these metrics, higher is better.

`TL`	**Trajectory Length** measures the average length of all the predicted navigation trajectories in meters.
`SR`	**Success Rate** measures the ratio of successful tasks, of which the agent's stop location is less than 3 meters away from the target location.
`OSR`	**Oracle Success Rate** measures the ratio of tasks of which one of its trajectory viewpoints can observe the target object within 3 meters.
`SPL`	**Success weighted by Path Length** trades-off SR (Success Rate) against TL (Trajectory Length). It measures both the accuracy and efficiency of navigation.
`RGS`	**Remote Grounding Success rate** measures the ratio of tasks that successfully locate the target object.
`RGSPL`	**RGS weighted by Path Length** is RGS.

Implementation Details

Our model is trained on a single 3090 GPU for 30,000 iterations. We set the batch size to 4 and the learning rate to 1×10^{-5}. The best model is selected according to performance on the validation unseen split. We use the same pretrained model and augmented data as [16] for a fair comparison. For the LLMs, we use the public GPT-2 [98] model for in-context learning. For the scene preceptor, we keep the top 3 object predictions for each position.

5.4.2 Comparison with State-of-The-Art Methods

As shown in Table 5.1, we compare MiC with the state-of-the-art methods on the REVERIE benchmark. Our method outperforms previous methods in all metrics on both validation unseen and test unseen splits. Particularly, compared with the SoTA method HM3D-DUET [16], MiC outperforms HM3D-DUET by a large margin of 3.09% in terms of the main navigation metric SPL and 3.49% of the main object grounding metric RGSPL on the Test Unseen split. Note that MiC shares the same pre-trained model with the HM3D-DUET, these promising result demonstrates that our method can effectively improve the navigation and object grounding ability of agents.

TABLE 5.1. Comparison with the state-of-the-art methods on REVERIE.

Methods	**Val Unseen**						**Test Unseen**					
		Navigation			**Grounding**			**Navigation**			**Grounding**	
	TL	OSR↑	SR↑	SPL↑	RGS↑	RGSPL↑	TL	OSR↑	SR↑	**SPL↑**	RGS↑	RGSPL↑
Human	–	–	–	–	–	–	21.18	86.83	81.51	53.66	77.84	51.44
Seq2Seq	11.07	8.07	4.20	2.84	2.16	1.63	10.89	6.88	3.99	3.09	2.00	1.58
RCM [122]	11.98	14.23	9.29	6.97	4.89	10.60	7.84	3.89	11.68	6.67	3.67	3.14
SMNA [76]	9.07	11.28	8.15	6.44	4.54	3.61	9.23	8.39	5.80	4.53	3.10	2.39
FAST-MATTN [95]	45.28	28.20	14.40	7.19	7.84	4.67	39.05	30.63	19.88	11.61	11.28	6.08
ORIST [93]	10.90	25.02	16.84	15.14	8.52	7.58	11.38	29.20	22.19	18.97	10.68	9.28
CKR [25]	26.26	31.44	19.14	11.84	11.45	-	22.46	30.40	22.00	14.25	11.60	-
RecBERT [39]	16.78	35.02	30.67	24.90	18.77	15.27	15.86	32.91	29.61	23.99	16.50	13.51
Airbert [29]	18.71	34.51	27.89	21.88	18.23	14.18	17.91	34.20	30.28	23.61	16.83	13.28
HAMT [15]	14.08	36.84	32.95	30.20	18.92	17.28	13.62	33.41	30.40	26.67	14.88	12.08
HOP [96]	16.46	36.24	31.78	26.11	18.85	15.73	16.38	33.06	30.17	24.34	17.69	14.34
TD-STP [133]	-	39.48	34.88	27.32	21.16	16.56	-	40.26	35.89	27.51	19.88	15.40
DUET [17]	22.11	51.07	46.98	33.73	32.15	23.03	21.30	56.91	52.51	36.06	31.88	22.06
HM3D-DUET [16]	-	62.14	55.89	40.85	36.58	26.76	-	62.30	55.17	38.88	32.23	22.68
MiC	20.64	**62.37**	**56.97**	**43.60**	**37.52**	**28.72**	**18.11**	**62.40**	**55.74**	**41.97**	**35.25**	**26.17**

TABLE 5.2. Ablation of different components in MiC.

Components	**Navigation**			**Grounding**	
	OSR↑	SR↑	SPL↑	RGS↑	RGSPL↑
HLI	58.02	52.71	40.49	34.93	26.82
HLI+GOSiP	59.92	55.28	42.46	37.13	28.24
HLI+SODiP	60.72	56.26	42.94	36.80	27.81
HLI+GOSiP+SODiP	62.37	56.97	43.60	37.52	28.72

5.4.3 Ablation Analysis

Contribution of different MiC Components

In Table 5.2, we evaluate the effect of different components in our proposed MiC. HLI denotes only using the original high-level instruction (HLI) provided by REVERIE.

Compared to the baseline HLI, GOSiP improves the performance of both navigation (2.57%↑ on SR, 1.97%↑ on SPL) and object grounding (2.20%↑ on RGS, 1.42%↑ on RGSPL) with a non-trivial margin, showing the effectiveness of the proposed goal-oriented static planning. SODiP further surpasses GOSiP in the navigation metric (0.98%↑ on SR, 0.48%↑ on SPL) while falling a little behind in the grounding metrics (0.33%↑ on RGS, 1.43%↑ on RGSPL). The reason may be that the detailed step-by-step planning occupies a large proportion compared to the target object in the input texts, which can bring noise for object grounding while improving the navigation performance. When combining all these components, the final performance gets further increased in all metrics, which surpasses the baseline with a large margin (4.26%↑ on SR, 3.11%↑ on SPL, 2.59%↑ on RGS and 1.9%↑ on RGSPL). The promising results here show that these components are complementary to each other.

TABLE 5.3. Comparison of different plan generation settings.

Methods	Navigation			Grounding	
	OSR↑	SR↑	SPL↑	RGS↑	RGSPL↑
Baseline	58.02	52.71	40.49	34.93	26.82
Static	60.24	55.35	41.74	36.30	27.03
Dynamic	60.72	56.26	42.94	36.80	27.81

TABLE 5.4. Human study of the prompt setting for Scene-Oriented Dynamic Planning.

Methods	Relevancy	Rationality
Scene-Oriented Dynamic Planning	2.06	1.93
- **w/o** Dynamic Demonstration	1.41	1.23
- **w/o** ROASeP	1.64	1.55

The Effect of the ROASeP

To evaluate the effectiveness of ROASeP used for the scene-oriented dynamic planning, we conduct another ablation study via whether incorporating the feedback from the ROASeP module on REVERIE validation unseen set. We report results in three settings: **(I)** Baseline: The input assembled instruction only contains the given high-level instruction in REVERIE. **(II)** Static: The input assembled instruction contains the REVERIE and fine-grained static instructions. The difference between fine-grained static instruction and scene-oriented dynamic instruction is that the static fine-grained instruction is generated without ROASeP. More specifically, the query prompt for the LLM to generate step-by-step planning is fixed at each timestep, which only consists of the given high-level instruction and the selected demonstration. **(III)** Dynamic: The input assembled instruction contains the high-level instruction in REVERIE and scene-oriented dynamic planning instruction. As shown in Table 5.3, in the static setting, the performance in all metrics is improved compared to the baseline, indicating the effectiveness of the LLM's rich world knowledge in fine-grained planning. In the dynamic setting, the performance is further improved with non-trivial margins, showing the effectiveness of ROASeP.

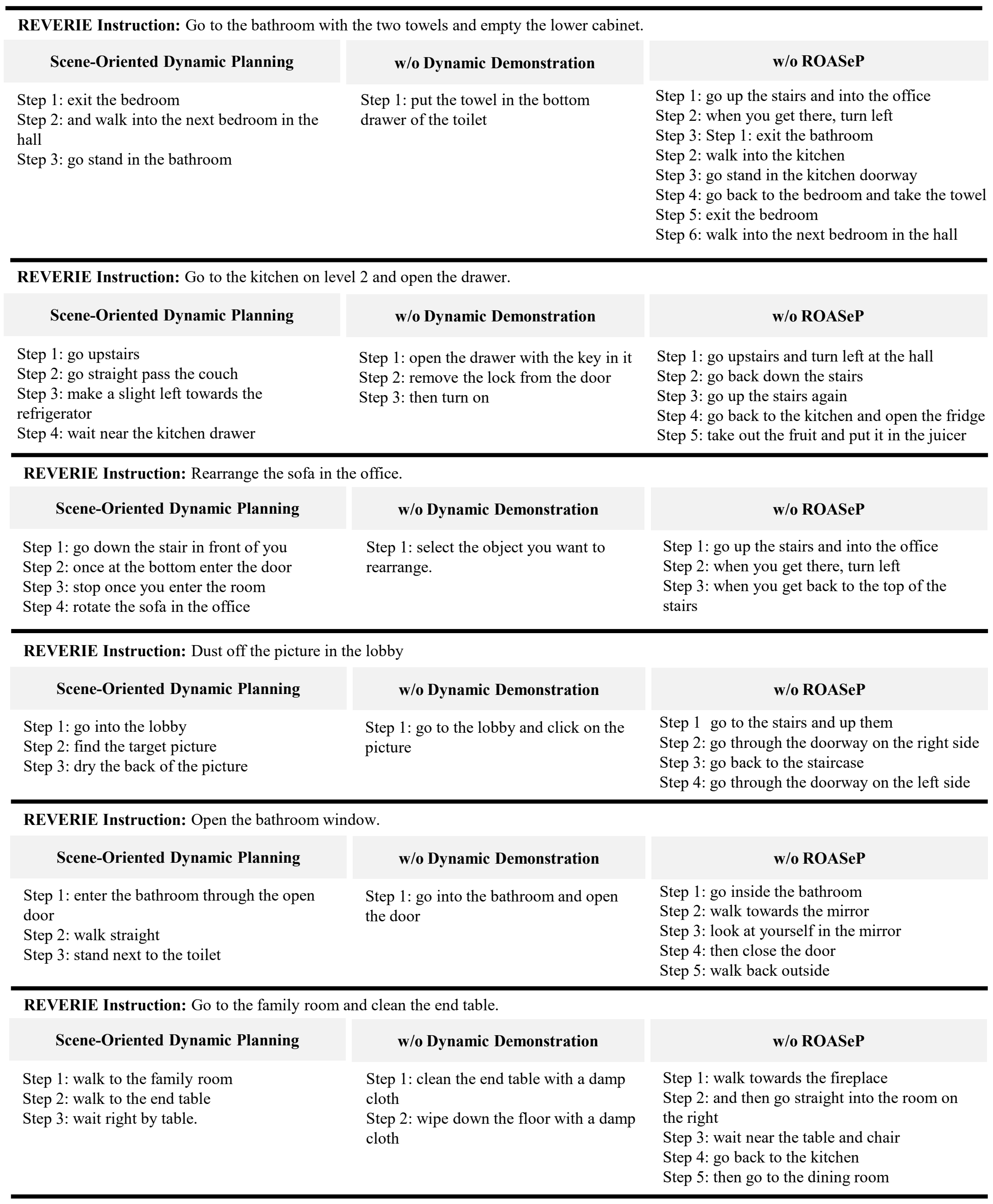
Qualitative Analysis of Prompt Setting

To further evaluate the effect of dynamic demonstration and ROASeP in SODiP, we perform a human evaluation of the generated plannings (see Table 5.4) and show the planning results (see Fig. 5.6). For human evaluation, we randomly selected 100 REVERIE tasks and generate fine-grained step-by-step instructions in the setting of SODiP, SODiP without dynamic demonstration, and SODiP without ROASeP. We asked 10 volunteers to mark the generated step-by-step instructions in terms of their relevancy and rationality. The relevancy score ranges from 0 (unrelated) to 3 (very related), which takes into account whether the keywords in instructions are related to the REVERIE task. For example, regarding the REVERIE instruction "Go to the kitchen and turn on the microwave", whether there are keywords in instructions related to the kitchen scene could be rated. Rationality is rated from 0 (bad) to 3 (perfect), considering whether the instruction conforms to the logic of navigation.

The results are presented in Table 5.4. It shows that our SODiP scored 2.06 on Relevancy and 1.93 on Rationality, which could be considered acceptable since the highest score is 3 and it is challenging to generate instructions that are consistent with tasks and actual navigation logic. When removing the dynamic demonstration, the score of generated instruction drops about 31.55% on Relevancy and 36.27% on Rationality, which could also be observed in Fig. 5.6. Although the instruction generated without dynamic demonstration is related to the task to some extent (*e.g.*, "put the towel in the bottom drawer of the toilet" has the keyword "towel", the instruction lacks navigation information, such as how to reach the bathroom.) As shown in the bottom example of Fig. 5.6, instruction without ROASeP successfully guided how to go to the destination location kitchen, but it still caused confusion by going upstairs and going downstairs several times, and thus reducing the rationality score, *i.e.* 1.64 on Relevancy and 1.55 on Rationality. More generation results can be found in the supplementary.

5.5 Conclusion

In this work, we propose a novel model, March-in-Chat (MiC), for the REVEIRE task, which only provides concise high-level instructions for the VLN agent. MiC enables the REVERIE agent to talk with an LLM on the fly to generate plans for the next few steps. It consists of three main modules, Goal-Oriented Static Planning (GOSiP), Scene-Oriented Dynamic Planning (SODiP), and Room-and-Object Aware Scene Perceiver (ROASeP) module. We conduct extensive quantitative and qualitative experiments on REVERIE and the promising results show the effectiveness of our method.



REVERIE Instruction: Go to the bathroom with the two towels and empty the lower cabinet.

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: exit the bedroom Step 2: and walk into the next bedroom in the hall Step 3: go stand in the bathroom	Step 1: put the towel in the bottom drawer of the toilet	Step 1: go up the stairs and into the office Step 2: when you get there, turn left Step 3: Step 1: exit the bathroom Step 2: walk into the kitchen Step 3: go stand in the kitchen doorway Step 4: go back to the bedroom and take the towel Step 5: exit the bedroom Step 6: walk into the next bedroom in the hall

REVERIE Instruction: Go to the kitchen on level 2 and open the drawer.

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: go upstairs Step 2: go straight pass the couch Step 3: make a slight left towards the refrigerator Step 4: wait near the kitchen drawer	Step 1: open the drawer with the key in it Step 2: remove the lock from the door Step 3: then turn on	Step 1: go upstairs and turn left at the hall Step 2: go back down the stairs Step 3: go up the stairs again Step 4: go back to the kitchen and open the fridge Step 5: take out the fruit and put it in the juicer

REVERIE Instruction: Rearrange the sofa in the office.

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: go down the stair in front of you Step 2: once at the bottom enter the door Step 3: stop once you enter the room Step 4: rotate the sofa in the office	Step 1: select the object you want to rearrange.	Step 1: go up the stairs and into the office Step 2: when you get there, turn left Step 3: when you get back to the top of the stairs

REVERIE Instruction: Dust off the picture in the lobby

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: go into the lobby Step 2: find the target picture Step 3: dry the back of the picture	Step 1: go to the lobby and click on the picture	Step 1 go to the stairs and up them Step 2: go through the doorway on the right side Step 3: go back to the staircase Step 4: go through the doorway on the left side

REVERIE Instruction: Open the bathroom window.

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: enter the bathroom through the open door Step 2: walk straight Step 3: stand next to the toilet	Step 1: go into the bathroom and open the door	Step 1: go inside the bathroom Step 2: walk towards the mirror Step 3: look at yourself in the mirror Step 4: then close the door Step 5: walk back outside

REVERIE Instruction: Go to the family room and clean the end table.

Scene-Oriented Dynamic Planning	**w/o Dynamic Demonstration**	**w/o ROASeP**
Step 1: walk to the family room Step 2: walk to the end table Step 3: wait right by table.	Step 1: clean the end table with a damp cloth Step 2: wipe down the floor with a damp cloth	Step 1: walk towards the fireplace Step 2: and then go straight into the room on the right Step 3: wait near the table and chair Step 4: go back to the kitchen Step 5: then go to the dining room

FIGURE 5.6. Examples of generated instructions.

Statement of Authorship

Title of Paper	March in Chat: Interactive Prompting for Remote Embodied Referring Expression.	
Publication Status	☐ Published	☐ Accepted for Publication
	☑ Submitted for Publication	☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Submitted to ICCV 2023	

Principal Author

Name of Principal Author (Candidate)	Yanyuan Qiao		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	28/06/2023

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yuankai Qi		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Zheng Yu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/06/2023

Name of Co-Author	Jing Liu

Contribution to the Paper	Discussion and writing revision.		
Signature		Date	29/06/2023

Name of Co-Author	Qi Wu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	29/06/2023

Chapter 6

Conclusion

In this chapter, we provide a comprehensive summary of our key contributions and propose potential research directions for future work in the field of Vision and Language Navigation (VLN).

The integration of vision and language holds tremendous potential for real applications, particularly in the area of Vision-and-Language Navigation (VLN). The VLN task involves enabling robots to understand navigation instructions expressed in natural language, perceive the environment, and execute corresponding actions, making it applicable in various scenarios such as home assistants.

While significant progress has been made in advancing the development of the VLN task, several challenges persist and require further attention. These challenges encompass three primary aspects: 1) The absence of a pre-trained model that specifically emphasizes temporal information for the VLN task. Existing models often focus on spatial understanding but lack dedicated temporal modeling tailored to the unique requirements of VLN; 2) The exploration of parameter-efficient transfer learning (PETL) for VLN tasks. Developing efficient transfer learning techniques that allow for effective knowledge transfer and fine-tuning while minimizing computational costs is an important area for future investigation; 3) The effective utilization of the rich knowledge provided by Large Language Models (LLMs) in VLN tasks. Harnessing the vast knowledge encapsulated within LLMs and integrating it effectively into VLN models could greatly enhance their decision-making capabilities.

First, we carefully examine and compare previous VLN methods, it has been observed that they often overlook the criticality of historical context in pre-training or fail to adequately consider the importance of action order in VLN tasks. To address this issue, a pre-training and fine-tuning paradigm with VLN-specific objectives was introduced, leveraging past observations to support future action prediction. Specifically, in addition to the commonly used Masked Language Modeling (MLM) and Trajectory-Instruction Matching (TIM) tasks, we introduce three novel VLN-specific proxy tasks: Action Prediction with History (APH) task, Trajectory Order Modeling (TOM) task and Group Order Modeling (GOM) task. The APH task incorporates the visual perception trajectory to enhance the learning of historical knowledge and improve action prediction. The TOM and GOM tasks focus on temporal visual-textual alignment, enhancing the agent's reasoning ability regarding the correct order of actions. Furthermore, we address the challenge of representation inconsistency in the historical context between the pre-training and fine-tuning stages by designing a memory network. This memory network effectively selects and summarizes historical information for action prediction during fine-tuning, mitigating the need for significant additional computational resources in downstream VLN tasks.

Additionally, our research delved into the exploration of Parameter-Efficient Transfer Learning (PETL) techniques, leading to the development of a VLN-specific PETL method. Our approach involves the design of two PETL modules: Historical Interaction Booster (HIB) and Cross-modal Interaction Booster (CIB). These modules are then combined with existing PETL methods to form the integrated VLN-PETL framework. This investigation represents the first study to apply PETL techniques specifically to VLN tasks, showcasing its efficacy in effectively tackling challenges within the domain.

Finally, the thesis also investigated the integration of utilization of Large Language Models (LLMs) and introduced the MiC model, which enables conversations with LLMs and dynamic planning based on the Room-and-Object Aware Scene Perceiver (ROASeP). The MiC model comprises three essential modules: Goal-Oriented Static Planning (GOSiP) module, Scene-Oriented Dynamic Planning (SODiP) module, and one Room-and-Object Aware Scene Perceiver (ROASeP) module. Through this approach, the potential of LLMs in enhancing VLN capabilities is demonstrated, as the MiC model effectively leverages LLMs to facilitate intelligent conversations and adaptive planning, leading to improved performance in VLN tasks.

6.1 Future Work

Based on the aforementioned research and findings, future work in the field of VLN has the following possibilities.

Parameter-Efficient Transfer Learning (PETL) techniques have shown great potential in addressing the challenges of VLN tasks. Future research could delve deeper into the design and optimization of PETL methods tailored for VLNs, aiming at a more efficient knowledge transfer and fine-tuning process. Especially in addition to reducing training parameters, it can also be studied from the perspective of reducing memory.

In addition, the utilization of Large Language Models (LLMs) in VLN tasks opens up new opportunities for research. Future work can focus on leveraging LLMs to facilitate more intelligent and interactive conversations with agents, enabling dynamic planning and adaptive decision-making in real-time scenarios. Exploring different strategies for integrating LLMs into VLN models and investigating the impact of various LLM architectures on navigation performance are potential directions for future exploration.

Finally, the application of VLN in real-world settings, such as smart homes or assistive robotics, presents exciting opportunities for future research. Investigating the deployment of VLN models in practical environments and addressing challenges such as human-robot interaction, robustness to dynamic surroundings, and long-term navigation are important areas to explore.

Bibliography

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. "Do As I Can and Not As I Say: Grounding Language in Robotic Affordances". In: *arXiv preprint arXiv:2204.01691.* 2022.

[2] Dong An, Yuankai Qi, Yan Huang, Qi Wu, Liang Wang, and Tieniu Tan. "Neighbor-view Enhanced Model for Vision and Language Navigation". In: *ACM Int. Conf. Multimedia.* 2021, pp. 5101–5109.

[3] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. "On Evaluation of Embodied Navigation Agents". In: *CoRR* abs/1807.06757 (2018).

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2018, pp. 6077–6086.

[5] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2018, pp. 3674–3683.

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. "VQA: Visual Question Answering". In: *Int. Conf. Comput. Vis.* 2015, pp. 2425–2433.

[7] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models". In: *ArXiv* abs/2106.10199 (2022).

[8] Rodney A. Brooks. “A robust layered control system for a mobile robot”. In: *IEEE J. Robotics Autom.* 2 (1986), pp. 14–23.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. “Language Models are Few-Shot Learners”. In: *Adv. Neural Inform. Process. Syst.* 2020, pp. 1877–1901.

[10] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *2017 International Conference on 3D Vision (3DV)* (2017), pp. 667–676.

[11] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *International Conference on 3D Vision, 3DV 2017, Qingdao, China.* 2017, pp. 667–676.

[12] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. “TOUCHDOWN: Natural Language Navigation and Spatial Reasoning in Visual Street Environments”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2019, pp. 12538–12547.

[13] Jinyu Chen, Chen Gao, Erli Meng, Qiong Zhang, and Si Liu. “Reinforced structured state-evolution for vision-language navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 15450–15459.

[14] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. “Evaluating large language models trained on code”. In: *arXiv preprint arXiv:2107.03374* (2021).

[15] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. “History Aware Multimodal Transformer for Vision-and-Language Navigation”. In: *Adv. Neural Inform. Process. Syst.* 2021, pp. 5834–5847.

[16] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. “Learning from Unlabeled 3D Environments for Vision-and-Language Navigation”. In: *Eur. Conf. Comput. Vis.* 2022, pp. 638–655.

[17] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. “Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 16516–16526.

[18] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. “UNITER: UNiversal Image-TExt Representation Learning”. In: *Eur. Conf. Comput. Vis.* 2020, pp. 104–120.

[19] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *ArXiv* abs/1601.06733 (2016).

[20] Ron Chrisley. “Embodied artificial intelligence”. In: *Artificial intelligence* 149.1 (2003), pp. 131–150.

[21] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. “RoboTHOR: An Open Simulation-to-Real Embodied AI Platform”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* (2020), pp. 3161–3171.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter Associat. Comput. Linguist.: Human Languag. Technol.* 2019, pp. 4171–4186.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *Int. Conf. Learn. Represent.* 2021.

[24] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. “Speaker-Follower Models for Vision-and-Language Navigation”. In: *Adv. Neural Inform. Process. Syst.* 2018, pp. 3318–3329.

[25] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. “Room-and-object aware knowledge reasoning for remote embodied referring expression”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2021, pp. 3064–3073.

[26] Lianli Gao, Xiangpeng Li, Jingkuan Song, and Heng Tao Shen. “Hierarchical LSTMs with Adaptive Attention for Visual Captioning”. In: 42.5 (2020), pp. 1112–1131.

[27] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[28] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. “Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions”. In: *Associat. Comput. Linguist.* 2022, pp. 7606–7623.

[29] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. "Airbert: In-Domain Pretraining for Vision-and-Language Navigation". In: *Int. Conf. Comput. Vis.* 2021, pp. 1634–1643.

[30] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. "Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-Training". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2020, pp. 13134–13143.

[31] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. "Towards a Unified View of Parameter-Efficient Transfer Learning". In: *Int. Conf. Learn. Represent.* 2022.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 42 (2017), pp. 386–397.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conf. Comput. Vis. Pattern Recog.* IEEE Computer Society, 2016, pp. 770–778.

[34] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. "On the Effectiveness of Adapter-based Tuning for Pretrained Language Model Adaptation". In: *ACL/IJCNLP*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. 2021, pp. 2208–2222.

[35] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. "Parameter-efficient Fine-tuning for Vision Transformers". In: *CoRR* abs/2203.16329 (2022).

[36] Yicong Hong, Cristian Rodriguez Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. "Language and Visual Entity Relationship Graph for Agent Navigation". In: *Adv. Neural Inform. Process. Syst.* 2020.

[37] Yicong Hong, Cristian Rodriguez Opazo, Qi Wu, and Stephen Gould. "Sub-Instruction Aware Vision-and-Language Navigation". In: *Proc. Conf. Empirical Methods Natural Lang. Process.* Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. 2020, pp. 3360–3376.

[38] Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. "Bridging the Gap Between Learning in Discrete and Continuous Environments for Vision-and-Language Navigation". In: *CVPR.* 2022, pp. 15418–15428.

[39] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez Opazo, and Stephen Gould. ": A Recurrent Vision-and-Language BERT for Navigation". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2021, pp. 1643–1653.

[40] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-efficient transfer learning for NLP". In: *Int. Conf. Mach. Learn.* 2019, pp. 2790–2799.

[41] Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models". In: *Int. Conf. Learn. Represent.* 2022.

[42] Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhães, Jason Baldridge, and Eugene Ie. "Transferable Representation Learning in Vision-and-Language Navigation". In: *Int. Conf. Comput. Vis.* 2019, pp. 7403–7412.

[43] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents". In: *Int. Conf. Mach. Learn.* PMLR. 2022, pp. 9118–9147.

[44] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. "Inner Monologue: Embodied Reasoning through Planning with Language Models". In: *arXiv preprint arXiv:2207.05608.* 2022.

[45] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. "General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping". In: *Adv. Neural Inform. Process. Syst.* 2019.

[46] Weston Jason, Chopra Sumit, and Bordes Antoine. "Memory networks". In: *arXiv preprint arXiv:1410.3916* (2014).

[47] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation". In: *Trans. Assoc. Comput. Linguistics* 5 (2017), pp. 339–351.

[48] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. "A New Path: Scaling Vision-and-Language Navigation with Synthetic Instructions and Imitation Learning". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2023, pp. 10813–10823.

[49] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha S. Srinivasa. "Tactical Rewind: Self-Correction via Backtracking in Vision-And-Language Navigation". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2019, pp. 6741–6749.

[50] Jing Yu Koh, Harsh Agrawal, Dhruv Batra, Richard Tucker, Austin Waters, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. "Simple and Effective Synthesis of Indoor 3D Scenes". In: *arXiv preprint arXiv:2204.02960* (2022).

[51] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Kumar Gupta, and Ali Farhadi. "AI2-THOR: An Interactive 3D Environment for Visual AI". In: *ArXiv* abs/1712.05474 (2017).

[52] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. "Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments". In: *Eur. Conf. Comput. Vis.* Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12373. Lecture Notes in Computer Science. Springer, 2020, pp. 104–120.

[53] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". In: *Int. J. Comput. Vis.* 123 (2016), pp. 32–73.

[54] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. "Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding". In: *Proc. Conf. Empirical Methods Natural Lang. Process.* Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. 2020, pp. 4392–4412.

[55] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. "Ask Me Anything: Dynamic Memory Networks for Natural Language Processing". In: *Int. Conf. Mach. Learn.* 2016, pp. 1378–1387.

[56] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. "RMA: Rapid Motor Adaptation for Legged Robots". In: *ArXiv* abs/2107.04034 (2021).

[57] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. "Stacked Cross Attention for Image-Text Matching". In: *Eur. Conf. Comput. Vis.* 2018, pp. 212–228.

[58] Brian Lester, Rami Al-Rfou, and Noah Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning". In: *Proc. Conf. Empirical Methods Natural Lang. Process.* 2021, pp. 3045–3059.

[59] Jialu Li, Hao Tan, and Mohit Bansal. "Envedit: Environment editing for vision-and-language navigation". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 15407–15417.

[60] Jialu Li, Hao Tan, and Mohit Bansal. "Improving Cross-Modal Alignment in Vision Language Navigation via Syntactic Information". In: *North American Chapter Associat. Comput. Linguist.: Human Languag. Technol.* 2021, pp. 1041–1050.

[61] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Çelikyilmaz, Jianfeng Gao, Noah A. Smith, and Yejin Choi. "Robust Navigation with Language Pretraining and Stochastic Sampling". In: *Proc. Conf. Empirical Methods Natural Lang. Process.* 2019, pp. 1494–1499.

[62] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. "Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks". In: *Eur. Conf. Comput. Vis.* 2020, pp. 121–137.

[63] Xiwen Liang, Fengda Zhu, Lingling Li, Hang Xu, and Xiaodan Liang. "Visual-Language Navigation Pretraining via Prompt-based Environmental Self-exploration". In: *Associat. Comput. Linguist.* 2022, pp. 4837–4851.

[64] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[65] Bingqian Lin, Yi Zhu, Zicong Chen, Xiwen Liang, Jianzhuang Liu, and Xiaodan Liang. "ADAPT: Vision-Language Navigation with Modality-Aligned Action Prompts". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 15396–15406.

[66] Bingqian Lin, Yi Zhu, Yanxin Long, Xiaodan Liang, Qixiang Ye, and Liang Lin. "Retreat for Advancing: Dynamic Reinforced Instruction Attacker for Robust Visual Navigation." In: *IEEE Trans. Pattern Anal. Mach. Intell.* PP (2021).

[67] Chuang Lin, Yi Jiang, Jianfei Cai, Lizhen Qu, Gholamreza Haffari, and Zehuan Yuan. "Multimodal Transformer with Variable-length Memory for Vision-and-Language Navigation". In: *ArXiv* abs/2111.05759 (2021).

[68] Xiangru Lin, Guanbin Li, and Yizhou Yu. "Scene-intuitive agent for remote embodied visual grounding". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2021, pp. 7036–7045.

[69] Chong Liu, Fengda Zhu, Xiaojun Chang, Xiaodan Liang, Zongyuan Ge, and Yi-Dong Shen. "Vision-Language Navigation with Random Environmental Mixup". In: *Int. Conf. Comput. Vis.* IEEE, 2021, pp. 1624–1634.

[70] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. "What Makes Good In-Context Examples for GPT-3?" In: *arXiv preprint arXiv:2101.06804* (2021).

[71] Yonatan Bisk Ari Holtzman Zhe Gan Jingjing Liu Jianfeng Gao Yejin Choi Siddhartha Srinivasa. Liyiming Ke Xiujun Li. “Tactical Rewind: Self-Correction via Backtracking in Vision-and-Language Navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2019, pp. 6741–6749.

[72] Siqu Long, Feiqi Cao, Soyeon Caren Han, and Haiqing Yang. “Vision-and-Language Pretrained Models: A Survey”. In: *ArXiv* abs/2204.07356 (2022).

[73] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *Int. Conf. Learn. Represent.* OpenReview.net, 2019.

[74] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks”. In: *Adv. Neural Inform. Process. Syst.* 2019, pp. 13–23.

[75] Chao Ma, Chunhua Shen, Anthony Dick, Qi Wu, Peng Wang, Anton van den Hengel, and Ian Reid. “Visual question answering with memory-augmented networks”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2018, pp. 6975–6984.

[76] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. “Self-Monitoring Navigation Agent via Auxiliary Progress Estimation”. In: *Int. Conf. Learn. Represent.* 2019.

[77] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. “The Regretful Agent: Heuristic-Aided Navigation Through Progress Estimation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2019, pp. 6732–6740.

[78] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. “Compacter: Efficient Low-Rank Hypercomplex Adapter Layers”. In: *Adv. Neural Inform. Process. Syst.* 2021, pp. 1022–1035.

[79] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. “Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks”. In: *ACL/IJCNLP*. 2021, pp. 565–576.

[80] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. “Improving Vision-and-Language Navigation with Image-Text Pairs from the Web”. In: *Eur. Conf. Comput. Vis.* 2020, pp. 259–274.

[81] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. “UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning”. In: *Associat. Comput. Linguist.* 2022, pp. 6253–6264.

[82] Bar Mayo, Tamir Hazan, and Ayellet Tal. “Visual Navigation with Spatial Attention”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* (2021), pp. 16893–16902.

[83] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: *Int. Conf. Mach. Learn.* Vol. 48. JMLR.org, 2016, pp. 1928–1937.

[84] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, Stefan Lee, and Dhruv Batra. "SOAT: A Scene-and Object-Aware Transformer for Vision-and-Language Navigation". In: *Adv. Neural Inform. Process. Syst.* 34 (2021).

[85] Tongzhou Mu, Z. Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. "ManiSkill: Generalizable Manipulation Skill Benchmark with Large-Scale Demonstrations". In: *NeurIPS Datasets and Benchmarks.* 2021.

[86] Andrew Y Ng, Stuart Russell, et al. "Algorithms for inverse reinforcement learning." In: *Int. Conf. Mach. Learn.* Vol. 1. 2000, p. 2.

[87] Yulei Niu, Hanwang Zhang, Zhiwu Lu, and Shih-Fu Chang. "Variational Context: Exploiting Visual and Textual Context for Grounding Referring Expressions". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.1 (2021), pp. 347–359.

[88] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. "ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning for Action Recognition". In: *CoRR* abs/2206.13559 (2022).

[89] Alexander Pashevich, Cordelia Schmid, and Chen Sun. "Episodic Transformer for Vision-and-Language Navigation". In: *Int. Conf. Comput. Vis.* IEEE, 2021, pp. 15922–15932.

[90] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: *ArXiv* abs/2004.00784 (2020).

[91] Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. "Synchromesh: Reliable Code Generation from Pre-trained Language Models". In: *Int. Conf. Learn. Represent.* 2022.

[92] Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. "Universal Dependency Parsing from Scratch". In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies.* 2018, pp. 160–170.

[93] Yuankai Qi, Zizheng Pan, Yicong Hong, Ming-Hsuan Yang, Anton van den Hengel, and Qi Wu. "The Road to Know-Where: An Object-and-Room Informed Sequential BERT for Indoor Vision-Language Navigation". In: *Int. Conf. Comput. Vis.* 2021, pp. 1655–1664.

[94] Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. “Object-and-Action Aware Model for Visual Language Navigation”. In: *Eur. Conf. Comput. Vis.* 2020, pp. 303–317.

[95] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. “REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2020, pp. 9979–9988.

[96] Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. “HOP: History-and-Order Aware Pre-training for Vision-and-Language Navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 8524–8537.

[97] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Int. Conf. Mach. Learn.* 2021, pp. 8748–8763.

[98] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[99] Santhosh K. Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. “PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* (2022), pp. 18868–18878.

[100] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. “Generative Adversarial Text to Image Synthesis”. In: *Int. Conf. Mach. Learn.* 2016.

[101] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *EMNLP-IJCNLP*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. 2019, pp. 3980–3990.

[102] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (), pp. 1137–1149.

[103] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *Int. J. Comput. Vis.* 115.3 (2015), pp. 211–252.

[104] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. “Habitat: A Platform for Embodied AI Research”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 9338–9346.

[105] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Associat. Comput. Linguist.* 2018, pp. 2556–2565.

[106] Bokui Shen, Fei Xia, Chengshu Li, Roberto Mart’in-Mart’in, Linxi (Jim) Fan, Guanzhi Wang, S. Buch, Claudia. Pérez D’Arpino, Sanjana Srivastava, Lyne P. Tchapmi, Micael Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. “iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes”. In: *International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 7520–7527.

[107] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. “ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2020, pp. 10737–10746.

[108] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. “ProgPrompt: Generating Situated Robot Task Plans using Large Language Models”. In: (2022). arXiv: `2209.11302 [cs.RO]`.

[109] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. “LLM-planner: Few-shot grounded planning for embodied agents with large language models”. In: *arXiv preprint arXiv:2212.04088* (2022).

[110] Robyn Speer, Joshua Chin, and Catherine Havasi. “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”. In: *ArXiv* abs/1612.03975 (2016).

[111] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. “End-To-End Memory Networks”. In: *Adv. Neural Inform. Process. Syst.* 2015, pp. 2440–2448.

[112] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. “LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning”. In: *CoRR* abs/2206.06522 (2022).

[113] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun,

Jitendra Malik, Manolis Savva, and Dhruv Batra. “Habitat 2.0: Training Home Assistants to Rearrange their Habitat”. In: *ArXiv* abs/2106.14405 (2021).

[114] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. “Habitat 2.0: Training home assistants to rearrange their habitat”. In: *Adv. Neural Inform. Process. Syst.* (2021), pp. 251–266.

[115] Hao Tan and Mohit Bansal. “LXMERT: Learning Cross-Modality Encoder Representations from Transformers”. In: *Proc. Conf. Empirical Methods Natural Lang. Process.* 2019, pp. 5099–5110.

[116] Hao Tan, Licheng Yu, and Mohit Bansal. “Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout”. In: *North American Chapter Associat. Comput. Linguist.: Human Languag. Technol.* 2019, pp. 2610–2621.

[117] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. “Vision-and-Dialog Navigation”. In: *Conf. on Robot Learn.* 2019, pp. 394–406.

[118] Hanqing Wang, Wenguan Wang, Wei Liang, Jianbing Shen, and Luc Van Gool. “Counterfactual Cycle-Consistent Learning for Instruction Following and Generation in Vision-Language Navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 15450–15460.

[119] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. “Active Visual Information Gathering for Vision-Language Navigation”. In: *Eur. Conf. Comput. Vis.* 2020.

[120] Hu Wang, Qi Wu, and Chunhua Shen. “Soft Expert Reward Learning for Vision-and-Language Navigation”. In: *Eur. Conf. Comput. Vis.* 2020, pp. 126–141.

[121] Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. “Less is More: Generating Grounded Navigation Instructions from Landmarks”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 15428–15438.

[122] Xin Wang, Qiuyuan Huang, Asli Çelikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. “Reinforced Cross-Modal Matching and Self-Supervised Imitation Learning for Vision-Language Navigation”. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2019, pp. 6629–6638.

[123] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. “Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation”. In: *Eur. Conf. Comput. Vis.* 2018, pp. 37–53.

[124] Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. "Visual question answering: A survey of methods and datasets". In: *Computer Vision and Image Understanding* 163 (2017), pp. 21–40.

[125] Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. "GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model". In: *Workshop on the Algorithmic Foundations of Robotics*. 2021, pp. 523–539.

[126] Caiming Xiong, Stephen Merity, and Richard Socher. "Dynamic Memory Networks for Visual and Textual Question Answering". In: *Int. Conf. Mach. Learn.* 2016, pp. 2397–2406.

[127] Mohit Bansal Yi-Lin Sung Jaemin Cho. "VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022, pp. 5217–5227.

[128] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L. Berg. "MAttNet: Modular Attention Network for Referring Expression Comprehension". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2018, pp. 1307–1315.

[129] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. "Modeling Context in Referring Expressions". In: *Eur. Conf. Comput. Vis.* 2016, pp. 69–85.

[130] Jiwen Zhang, Jianqing Fan, Jiajie Peng, et al. "Curriculum learning for vision-and-language navigation". In: *Adv. Neural Inform. Process. Syst.* 34 (2021), pp. 13328–13339.

[131] Yue Zhang and Parisa Kordjamshidi. "LOViS: Learning Orientation and Visual Signals for Vision and Language Navigation". In: 2022, pp. 5745–5754.

[132] Zhengkun Zhang, Wenya Guo, Xiaojun Meng, Yasheng Wang, Yadao Wang, Xin Jiang, Qun Liu, and Zhenglu Yang. "Hyperpelt: Unified parameter-efficient language model tuning for both language and vision-and-language tasks". In: *arXiv preprint arXiv:2203.03878* (2022).

[133] Yusheng Zhao, Jinyu Chen, Chen Gao, Wenguan Wang, Lirong Yang, Haibing Ren, Huaxia Xia, and Si Liu. "Target-Driven Structured Transformer Planner for Vision-Language Navigation". In: *ACM Int. Conf. Multimedia.* 2022, pp. 4194–4203.

[134] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Places: A 10 million image database for scene recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.6 (2017), pp. 1452–1464.

[135] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. "SOON: Scenario Oriented Object Navigation With Graph-Based Exploration". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2021, pp. 12689–12699.

[136] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. "Vision-Language Navigation With Self-Supervised Auxiliary Reasoning Tasks". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2020, pp. 10009–10019.

[137] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. "DM-GAN: Dynamic Memory Generative Adversarial Networks for Text-To-Image Synthesis". In: *IEEE Conf. Comput. Vis. Pattern Recog.* (2019), pp. 5795–5803.

[138] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. "BabyWalk: Going Farther in Vision-and-Language Navigation by Taking Baby Steps". In: *Associat. Comput. Linguist.* 2020, pp. 2539–2556.

[139] Yi Zhu, Fengda Zhu, Zhaohuan Zhan, Bingqian Lin, Jianbin Jiao, Xiaojun Chang, and Xiaodan Liang. "Vision-Dialog Navigation by Exploring Cross-Modal Memory". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2020, pp. 10727–10736.