天津大学

本科生毕业论文

题目：智能无人系统动态行为规划研究

学　　院　机械工程学院

专　　业　机械设计制造及其自动化

年　　级　2018 级

姓　　名　李泽康

学　　号　3018201223

指导教师　刘若楠

独创性声明

本人声明：所呈交的毕业设计（论文），是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业设计（论文）中不包含任何他人已经发表或撰写过的研究成果。对本毕业设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在论文中作了明确的说明。本毕业设计（论文）原创性声明的法律责任由本人承担。

论文作者签名：李泽康

2022 年 6 月 5 日

本人声明：本毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过论文的全部内容。

论文指导教师签名：刘若楠

2022 年 6 月 5 日

摘　要

一个能清楚理解人类语言并在现实视觉世界中进行智能导航的机器人可以为人类执行特定的任务，如桥梁检查、消防救火等。研究者希望使机器人能够理解自然语言指令，并在现实环境中结合视觉信息，执行相应动作到达指定目的地。这种导航任务就称为视觉语言导航。

本文针对视觉语言导航中的房间到房间任务，以 follower 模型为基础，研究并改进为 contrast-visionVAE-follower 模型，通过增加跨模态对比学习模块，学习语言和视觉跨模态信息的匹配关系，通过增加视觉变分自编码器模块，在有限的训练数据情况下，进行视觉信息编码与重建，增加训练时数据的多样性，提高模型在未见过的视觉环境中的泛化性能。实验结果显示 contrast-visionVAE-follower 模型在验证数据集上的导航成功率高于 follower 模型，且导航误差更小，具有更加优异的导航性能。

关键词：视觉语言导航，多模态学习，循环神经网络

ABSTRACT

A robot that can clearly understand human language and conduct intelligent navigation in the real visual world can perform specific tasks for human beings, such as bridge inspection, fire fighting and so on. Researchers hope that the robot can understand natural language instructions, and perform corresponding actions to reach the designated destination in the real environment combined with visual information. This navigation task is called Visual-and-Language Navigation.

Aiming at the room to room task in Visual-and-Language Navigation, based on the follower model, this paper studies and improves it into the contrast-visionVAE-follower model. By adding the cross modal contrastive learning module to learn the matching relationship between language and visual cross modal information, and by adding the visual variational auto-encoder module, the visual information is encoded and reconstructed under the condition of limited training data, so as to increase the diversity of data during training and improve the generalization performance of the model in an unprecedented visual environment. The experimental results show that the navigation success rate of contrast-visionVAE-follower model on the validation data set is higher than that of follower model, and the navigation error is smaller, which has better navigation performance.

KEY WORDS: Vision-and-Language Navigation, Multimodal Learning, Recurrent Neural Network

目 录

第一章　绪论

1.1　研究背景

机器人学和人工智能等相关领域的一个长期共同目标是研究出一种智能化的机器人，可以向其发送类似人类书面表达或口头描述的自然语言指令或句子，这些指令中包含特定的任务要求，机器人在接受到指令后，应能在给定的任务环境中执行所要求的的任务。尽管相关领域到目前为止已经有了许多先进的技术，然而要实现这样智能化的机器人仍然存在着许多重大的挑战。

在实现能执行复杂任务的机器人这一终极目标前，研究人员首先关注更为简单的任务：使机器人能够理解自然语言指令，并在现实环境中结合视觉信息，执行相应动作到达指定目的地。这种简单的导航任务就被称为视觉语言导航（Vision-and-Language Navigation，VLN）[1]。VLN 是将语言、视觉和导航在非结构化、不可见的环境中连接起来的一项非常重要的任务，随着相关领域的研究和技术的发展，目前已经引起了越来越多计算机视觉和自然语言处理领域研究者的关注。

然而，即使是简单的 VLN 任务，仍然存在许多技术难点，一是如何使得机器人能够学会在非结构化的、从未见过的环境中将自然语言指令、视觉环境和动作联系起来并进行合理的理解。二是直接在真实的现实环境中去训练机器人的难度较大，通常需要在计算机和模拟器上进行仿真分析，然而模拟器上的仿真常常使用的是渲染图像而非真实图像，这会将丰富多样的现实世界的可视对象集简单地约束为渲染器可用的有限的虚拟模型集，将 VLN 任务从一个具有挑战性的机器人开放集问题变成一个简单的封闭集分类问题。如果只是简单地使用基于渲染图像的模拟器进行仿真，则会导致训练出来的机器人的泛化性能下降，即机器人在模拟器中的表现良好，而在现实环境中则表现不佳。因此，使用真实世界的图像数据与虚拟图像数据相比，能极大地保留视觉和语言的丰富性，最大限度地提高在模拟器上训练有素的机器人转移到真实世界进行应用部署的潜力。

1.2　研究意义

机器人作为“中国制造 2025”十大领域之一，正迎来快速发展的黄金时期，机器人的智能化、自主化得到了国内外学者及工业届的普遍关

注，随着社会的快速发展，人们对智能机器人的需求也日益增加。人类在日常生活中常常需要使用自然语言进行相互交流、发布任务并向他人寻求帮助等。从现实意义来说，一个能清楚理解人类语言并在现实世界中进行智能导航的机器人将为人类社会带来许多巨大的好处。机器人和机器人之间可以像人类一样使用自然语言进行交谈，并且可以在没有人类监督和操控的情况下自主地执行人类给定的特定任务，例如进行室内家务活、室外重复性的体力劳动工作，或者是在较为危险的环境条件下依据人类预先给定的命令进行桥梁检查、消防救火等工作。从科学研究意义来说，研究和开发这样的机器人探索了基于人工智能的机器人如何解释人类的自然语言、感知其视觉环境并利用这些综合信息进行导航并成功完成人类给定的特定任务的潜能，不但能够促进人工智能领域中的如计算机视觉和自然语言处理领域的共同发展，还将影响传统的机器人学领域的研究。出色地执行 VLN 任务的能力将是进一步实现各种更高级的复杂任务的基础。

目前，VLN 仍是一个方兴未艾的研究领域。自房间到房间（Room-to-Room，R2R）数据集发布以来，许多关于 VLN 任务的数据集和研究工作已经被提出，然而，VLN 任务中任然存在着许多挑战。首先，机器人需要在一个复杂的视觉环境中学会如何有效地理解来自不同模态的信息。第二，机器人需要在导航过程中采用合理的推理策略以成功地到达目的地。第三，数据稀缺也是一个障碍，通常无法收集到大量的真实数据。最后，如何将在可见环境中训练的模型部署到不可见环境中也是一大挑战。

综合来看，尽管已经进行了一定的简化，VLN 仍然是一项较为困难的任务，它需要机器人能够理解来自人类的自然语言指令，编码来自周围现实环境的视觉信息，并将现实场景中的关键视觉特征和适当的动作与指令联系起来，以实现特定的目标（通常只是单纯的在现实环境中从起始位置移动到目标位置）。为了完成 VLN 任务，机器人需要学习对齐语言语义和视觉信息，并理解视觉和语言在交互过程中的动态变化。因为很难收集到足够的导航路径与人类注释指令的真实数据，VLN 任务的数据短缺问题使得在交互环境中学习视觉和语言之间的最佳匹配变得相当具有挑战性。同时，由于自然语言是一种抽象度高、信息密度大的数据，通常只包含少数高级的决策和地标，而不是完整的低级运动行为，所以许多缺失的信息必须由机器人综合视觉环境进行合理的推断。

因此，围绕当前各领域对智能化机器人的迫切需求，本课题针对视觉语言导航的房间到房间任务中的现存问题进行了系统研究。

1.3 国内外研究现状

在视觉和语言导航（VLN）任务中，房间到房间（Room-to-Room，R2R）任务是最先被研究者关注到的，VLN 的大部分研究工作都是基于 R2R 任务和 R2R 数据集展开的。这些工作的基本框架是序列到序列模型（Sequence-2-Sequence Model），它们侧重于采用不同的创新方法来提高模型的性能。

1.3.1 探索策略

对探索策略展开研究的相关工作的重点是使模型可以寻找到一条从起点到终点的有效且快速的路径。Ma 等人[2]提出了一个模块来评估 agent 前往目标的整个完整过程。在此基础上，Ma 等人[3]设计了用于前进或后退的“后悔”模块和帮助 agent 决定下一步方向的“进度标记”模块。虽然目前的所有方法都采用波束搜索来做出局部动作决策或对整个轨迹进行评分，但 Ke 等人[4]提出了具有回溯导航器的边界感知搜索（FAST）方法。当 agent 发现自己迷路时，FAST 导航器可以使用异步搜索进行显式回溯。Huang 等人[5]定义了两个辅助子任务：跨模态对齐（CMA）和下一视觉场景（NVS），使得在特定环境中学习的 agent 的视觉和文本表示可以转移到其他环境中。Zhu 等人[6]引入了四个自监督辅助推理任务，以利用从语义信息中获得的额外训练信号，帮助 agent 理解环境和任务，从而达到提高模型性能的效果。

1.3.2 数据增强

Fried 等人[7]提出了 speaker-follower 模型，用于监督学习中的数据增强和推理。Hong 等人[8]用子指令及其相应的路径丰富了基准 R2R 数据集，可以更好地在训练过程中为 agent 提供足够的语义信息。Agarwal 等人[9]提出了一个“正在进行”的 speaker 模型，该模型分两步生成导航指令，首先，使用硬注意力机制沿轨迹选择一系列离散的视觉地标，然后生成以这些地标为条件的语言。与传统的数据增强方法不同，Parvaneh 等人[10]提出了一种高效的算法来生成反事实实例并添加到训练过程中，从而提高了 agent 在新环境中的导航能力。反事实思想还有另一个应用，有研究者[11]引入了一种称为对抗路径采样器的模型不可知方法对路径进行采样，以逐步优化 agent 的导航策略。Yu 等人[12]处理了 R2R 任务中数据的稀缺性，同时通过随机游走数据增强的方式消除了数据集中的偏差，通过这样做，他们减少了泛化误差，使得模型在未知环境中的性

能优于基线模型。An 等人[13]设计了一个名为邻居视图增强模型的模块，用于在全局和局部级别自适应地融合来自邻居视图的视觉上下文。Liu 等人[14]提出了随机环境混合方法，将环境和相应的路径分解重组，构建一个全新的环境作为训练数据，缩小可见和不可见环境之间的性能差距，并提高整体性能。Sun 等人[15]指出，深度图像作为导航的一个有价值的信号源尚未得到充分探索，因此在以前的研究中常常被忽略。因此，他们提出了一个基于深度图像的自适应实例规范化模块和转移注意力模块来解决这个问题。

1.3.3 预训练模型

通过预训练模型得到的一般特征表示可以应用于各种下游任务，这种方法已经在许多研究领域得到了验证。一个强大的预训练骨干网络可以有效地改善下游任务的性能，如计算机视觉领域的图像识别任务和自然语言处理领域的问答任务。在 VLN 领域，Li 等人[16]通过预训练的语言模型（BERT [17]和 GPT-3[18]）和随机抽样对 agent 进行训练，以提高模型在不可见环境中的泛化性能。Hao 等人[19]开发了一种基于视觉和语言的通用预训练导航器，它是通过图像、语言和动作三元组进行预训练的，并在 R2R 任务上进行微调，以提升模型在未知环境中的泛化性能。由于 VLN 任务中训练数据的稀缺性，Majumdar 等人[20]试图使用大量的从网络上爬取的数据来解决这个问题，提出 VLN-BERT 模型，对网络中的图像文本对进行预训练然后在 R2R 任务上微调以显著提高 VLN 任务的性能。Hong 等人[21]提出了一种多模态 BERT 模型 VLN↻BERT，该模型具有时间感知的递归函数，为 agent 提供更丰富的信息。Qi 等人[22]提出了一种基于对象和房间的顺序 BERT 模型 ORIST，通过在相同细粒度级别（即对象和单词）编码视觉和指令输入的方式来提高语言落实性能。训练后的模型能够识别出各导航位置的相对方向以及当前和最终导航目标的房间类型。

1.4 论文研究内容

本文针对视觉和语言导航（VLN）中的房间到房间（R2R）任务，基于 speaker-follower 模型中用于导航的 follower 模型，研究并改进了一种基于全景视觉空间和高级决策动作的智能体导航模型，称为 contrast-visionVAE-follower 模型。该模型采用序列到序列模型结构，使用注意力机制学习自然语言指令和视觉图像序列

的对齐关系。引入跨模态对比学习模块和对比学习损失，学习语言和视觉跨模态信息的匹配关系，拉近相似跨模态信息在隐空间中的距离，拉远不相似跨模态信息在隐空间中的距离，使模型能更有效地编码语言信息和视觉信息。引入了视觉变分自编码器模块（visionVAE）和视觉信息重建损失，学习输入数据样本的视觉图像序列的概率分布，先编码真实视觉图像特征表示，再从视觉变分自编码器模块的编码隐空间概率分布中解码出相似但不完全相同的新视觉图像特征表示，使得在有限的训练数据情况下，通过视觉信息重建，增加训练时数据的多样性，提高模型在未见过的视觉环境中的泛化性能。实验结果显示 contrast-visionVAE-follower 模型在可见验证数据集和不可见验证数据集中的导航成功率与 follower 模型相比均有所提高，且导航误差也有所减小。

1.5 论文组织结构

本文共分为五个部分，组织结构如下：

第一章主要讲述了视觉语言导航（VLN）的研究背景和意义，概述了相关研究者在视觉语言导航领域的房间到房间（R2R）任务中的研究现状，并介绍了论文的研究内容和组织结构。

第二章是相关理论概述，首先对循环神经网络和长短期记忆网络的相关概念进行了介绍，然后介绍了本文参考的 speaker-follower 模型的具体结构和注意力机制，给出了全景动作空间的定义。

第三章重点介绍本文的改进模型，首先对具体要进行的 R2R 任务给出了具体的描述，然后介绍了改进的 contrast-visionVAE-follower 模型的具体结构，主要包括视觉变分自编码器模块和跨模态对比学习模块。同时还简单介绍了变分自编码器模型和对比学习模型。

第四章是实验结果与分析，首先给出了实验所使用的 R2R 数据集和 Matterport3D 模拟器的详细说明，然后介绍了实验环境的配置过程，最后给出了实验结果的评估指标以及实验结果分析。

第五章对本文的研究工作进行总结，并展望未来的研究工作。

第二章　相关理论概述

神经网络（Neural Network）是深度学习（Deep Learning）中使用广泛的模型，研究者使用神经网络进行图像分类、数值预测等任务，然而，传统的神经网络通常只能处理一维的数据点信息，而不能处理序列信息，因此学者们又提出了循环神经网络（Recurrent Neural Network，RNN）。但是，原始结构的 RNN 只能处理简单的序列建模问题，对于复杂的序列建模问题，它存在较为严重的记忆衰减问题，这将导致网络在训练过程中无法高效地学习较长序列信息中的重要部分。长短期记忆网络（Long Short Time Memory，LSTM）[23]正是为了解决这一问题而设计的一种特殊的 RNN 结构。

2.1　循环神经网络

循环神经网络（Recurrent Neural Network，RNN）对于处理具有序列特性的数据非常有效，与传统的全连接神经网络相比，循环神经网络能更好地挖掘和理解数据中的序列信息。序列特性即符合时间顺序、先后顺序或其他逻辑顺序的数据特性。生活中的序列数据有很多，如人类在交流时说出的句子、音乐播放器播放的音乐以及每天的股票价格变化等。

与全连接神经网络相比，循环神经网络最明显的变化是引入了一个称为“隐状态”（Hidden State）的概念。在当前时间步t的运算中，输入到循环神经网络中的除了当前时刻的原始输入x_t外，还有上一时刻的隐状态h_{t-1}，两者经过隐藏层的计算后又会生成当前时刻的隐状态h_t。循环神经网络基本结构如图 2-1 所示。

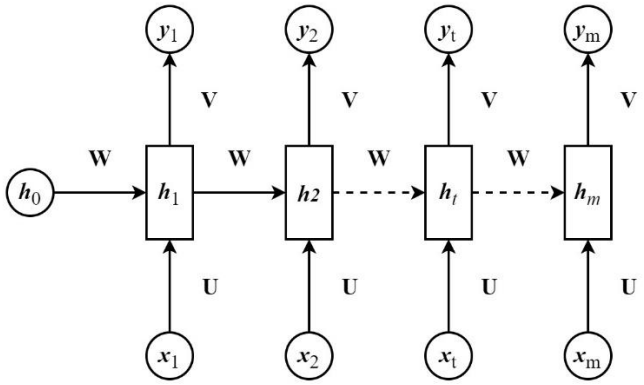

图 2-1　循环神经网络基本结构图

在时间步t，循环神经网络基本单元的数据运算过程如式（2-1）和式（2-2）所示：

$$h_t = f(\mathrm{U}x_t + \mathrm{W}h_{t-1} + b_1) \quad (2-1)$$

$$y_t = g(\mathrm{V}h_t + b_2) \quad (2-2)$$

其中x_t是当前时间步的输入，h_{t-1}是上一时间步的隐状态，h_t是当前时间步计算出的隐状态，y_t是当前时间步的输出，U、W、V分别是对应相乘的权重矩阵，b_1、b_2是偏置项，$f(\cdot)$和$g(\cdot)$是对应的激活函数。在每一时间步的运算中，所使用的参数U、W、V、b_1、b_2是一模一样的，也就是说，每一次计算的参数都是共享的，这是循环神经网络的一个重要特点。理论上，如果没有限制循环神经网络的时间步数，那么模型的序列化计算过程可以无限地进行下去。

2.2 长短期记忆网络

RNN 的特点之一是可以处理序列化信息，通过使用过去的历史信息来理解当前时刻的数据信息。然而，实际应用发现，一旦上下文信息与当前预测位置之间的距离增大时，RNN 会很快丧失掉学习到连接远距离上下文信息相互关系的能力。当输入的序列信息非常长时，RNN 可能在一开始就会遗漏掉重要信息，使得梯度随着时间步增加而迅速衰减，RNN 的学习能力随之下降，出现梯度消失的严重问题。这将导致 RNN 丢失长期记忆性，称为长期记忆依赖问题。为此，研究者在传统的 RNN 结构上，提出了一些著名的改进方案，其中之一就是长短期记忆网络（LSTM）。

LSTM 由 Hochreiter 和 Schmidhuber 在 1997 年提出，并被 AlexGraves 等人改良和推广。LSTM 经过精细的设计来避免长期记忆依赖问题，核心概念是细胞状态和“门”结构，同时保留了传统 RNN 的隐状态。细胞状态在 LSTM 中呈一条水平线贯穿左右，它的作用类似于传递物资的不可中断的公路，可以让信息在其上流传且保持不变。“门”结构就像一个个闸口，控制“门”结构的打开程度可以决定应该将哪些不重要的信息从细胞状态中除去或将更重要的信息增加到细胞状态中。通过细胞状态和各个“门”结构的相互配合，LSTM 可以使得较早的历史序列信息在经过长距离的传输后仍能在当前时间步的细胞状态中保留下来，这克服了传统 RNN 的短时记忆的缺点，使得 LSTM 具备更强更好的长期记忆能力。LSTM 模型的结构如图 2-2 所示。

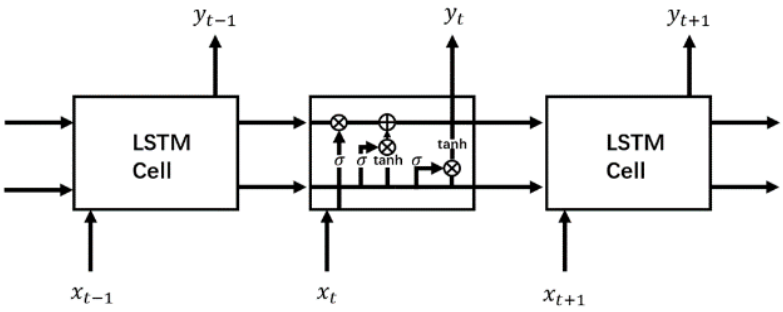

图 2-2　LSTM 模型结构图

下面将 LSTM 基本单元的结构图进行分解，逐步分析数据的处理和运算过程。

2.2.1　遗忘门

遗忘门的作用是决定从细胞状态中丢弃掉哪些信息。遗忘门结构的位置如图 2-3 所示。

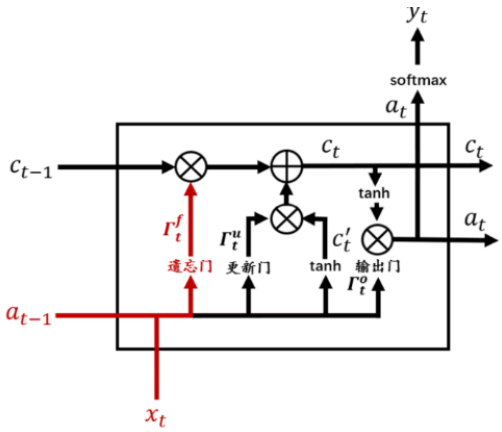

图 2-3　LSTM 基本单元中的遗忘门

遗忘门接受当前时间步t的原始输入和上一时间步$t-1$的隐状态，将原始输入和隐状态进行横向拼接后作加权运算，再经过一个 sigmoid 函数处理，得到数值范围在[0,1]之间的遗忘矩阵，用于后续计算。遗忘门的计算如式（2-3）所示：

$$\Gamma_t^f = \sigma\left(\mathrm{W}_f[a_{t-1}, x_t] + b_f\right) \qquad (2-3)$$

其中x_t为当前时间步的原始输入，a_{t-1}为上一时间步的隐状态，[,]为向量拼接操作，W_f为权重矩阵，b_f为偏置项，$\sigma(\cdot)$为 sigmoid 函数。Γ_t^f为遗忘矩阵。

2.2.2 记忆细胞候选值与更新门

更新门决定将哪些重要的信息存入细胞状态中。同时还要计算出要存入细胞状态的候选信息。更新门结构的位置如图 2-4 所示。

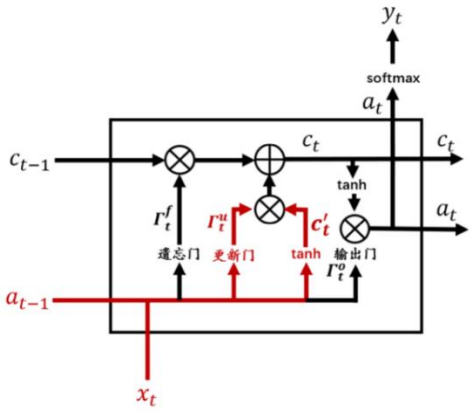

图 2-4 LSTM 基本单元中的更新门

更新门和细胞状态候选值的计算如式（2-4）和式（2-5）所示：

$$\Gamma_t^u = \sigma(\mathrm{W}_u[a_{t-1}, x_t] + b_u) \qquad (2-4)$$

$$c_t' = \tanh(\mathrm{W}_c[a_{t-1}, x_t] + b_c) \qquad (2-5)$$

其中W_u、W_c为相对应的权重矩阵，b_u、b_c为相对应的偏置项，$\tanh(\cdot)$为双曲正切函数，可以将输入值映射到$[-1,1]$的实数范围内。Γ_t^u为更新矩阵，c_t'为记忆细胞候选值。

2.2.3 记忆细胞更新

通过遗忘门、更新门计算出遗忘矩阵、更新矩阵和记忆细胞候选值后，便可以结合这些数据和上一时间步记忆细胞值计算出当前时间步的新的细胞状态，如式（2-6）所示：

$$c_t = \Gamma_t^f c_{t-1} + \Gamma_t^u c_t' \qquad (2-6)$$

其中c_t表示当前时间步的细胞状态，Γ_t^f、Γ_t^u、c_{t-1}和c_t'分别是之前已经计算出的遗忘矩阵、更新矩阵、上一时间步细胞状态和当前时间步记忆细胞候选值。

2.2.4 输出门与隐状态更新

LSTM 提供了单独的输出门用于计算模型的输出值和隐状态。输出门结构的位置如图 2-5 所示。

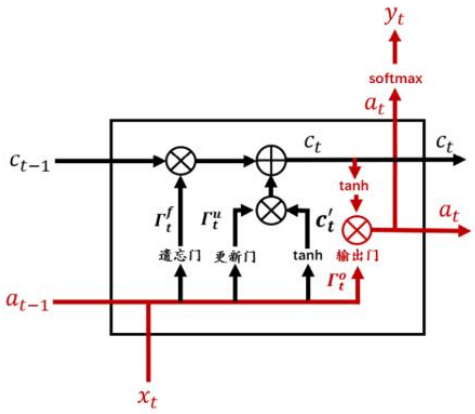

图 2-5　LSTM 基本单元中的输出门

输出门的计算如式（2-7）和式（2-8）所示：

$$\Gamma_t^o = \sigma(\mathrm{W}_o[a_{t-1}, x_t] + b_o) \qquad (2-7)$$

$$a_t = \Gamma_t^o \tanh(c_t) \qquad (2-8)$$

其中W_o为权重矩阵，b_o为偏置项，Γ_t^o为隐状态更新矩阵，a_t为当前时间步隐状态。y_t为最终的输出值，一般可以通过对a_t做 softmax 计算得到。

以上便是完整的 LSTM 基本单元的结构和计算过程。

2.3　Speaker-Follower 模型

本论文使用的模型基于 speaker-follower 模型[7]中的 follower 模型进行改进，follower 模型内嵌于全景视觉空间表示，可以直接进行高级动作控制。下面介绍全景动作空间和 follower 模型的具体结构。

2.3.1　全景动作空间

最基础的模型中的智能体（agent）通常采用低级视觉动作进行运动控制（例如向左或向右旋转相机 30 度），agent 每次只能从相机中看到它正前方的视觉场景信息，其他角度的视觉信息是看不见的。这种划分精细的视觉动作控制方式和受限的视觉观察信息虽然符合大多数生物的视觉特点，但是会给 agent 在执行导航任务时带来非常巨大的挑战。例如，如果输入的自然语言指令是英文句子“turn left and go towards the sofa”（中文意思：“向左转并向沙发走去”），那么 agent 就要执行一系列的转向操作，直到在相机视野中看到目标沙发，才能执行下一个“向前走”的动作，这需要 agent 具备非常强大的决策能力和对视觉环境信息的超强记忆。在 speaker-follower 模型中，直接允许 agent 执行高级视觉动作，即 agent 每次都能观察到全景表示的完整全景视觉空间信息，然后根据决策选择下

一个应该前往的位置。

如图 2-6 所示，与低级视觉运动空间相比，全景动作空间允许 agent 完全感知周围视觉信息并直接执行高级动作。在全景视觉空间中，agent 在每次执行动作前都会先环顾四周，把它周围环境的 360 度全景视觉信息全部观察一遍。为了处理方便，全景视觉空间被离散化为 36 个视角（其中水平旋转按航向角均分为 12 份，向前、向上和向下按仰角分为 3 份，以 30 度划分）。之后离散化的每个视角i都会被编码成对应的编码向量v_i。当 agent 处于某一位置时，由于场景中的某些物体的阻挡，agent 只能选择没有被阻挡的方向进行移动。agent 在每一位置的可导航方向j也会被编码成相应的编码向量u_j（注意，u_j和v_i不相同，v_i是客观存在的真实全景视觉空间编码信息，u_j则是全景空间中可以前往的无障碍的方向的编码信息）。全景视觉空间中的视觉图像特征使用已经预训练完成的 ConvNet 模型进行提取。agent 在环境中处于某一确定位置时还可以获得一个四维方向特征向量$[\sin\psi, \cos\psi, \sin\theta, \cos\theta]$，其中$\psi$是航向角，$\theta$是仰角。最终得到的编码向量$v_i$和$u_j$由视觉图像特征和四维方向特征拼接起来。为了使得 agent 在执行导航任务时可以自行决定何时停止，引入一个特定的“STOP”动作编码向量，用$u_0 = 0$表示。当 agent 认为其自身已经到达目的地后，可以执行该“STOP”动作停止导航任务。

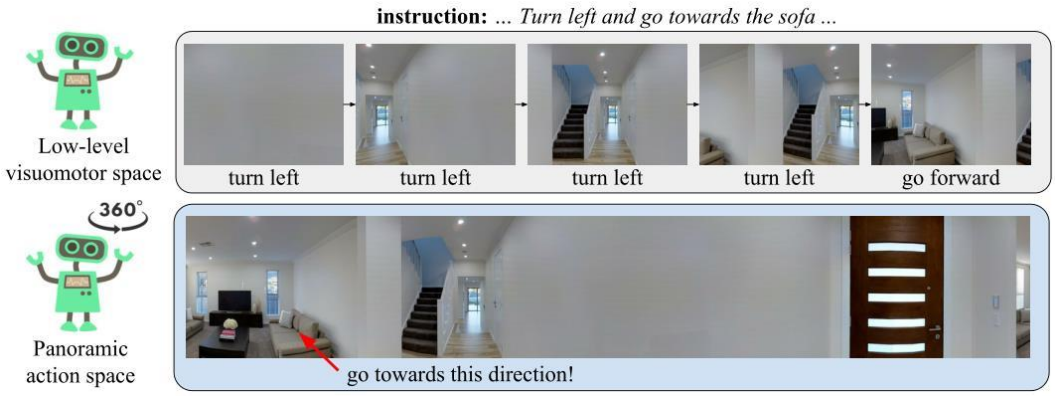

图 2-6　低级视觉动作空间与全景视觉动作空间对比图[7]

2.3.2 Follower 模型结构

follower 模型采用序列到序列结构，由编码器和解码器两大部分组成，如图 2-7 所示。

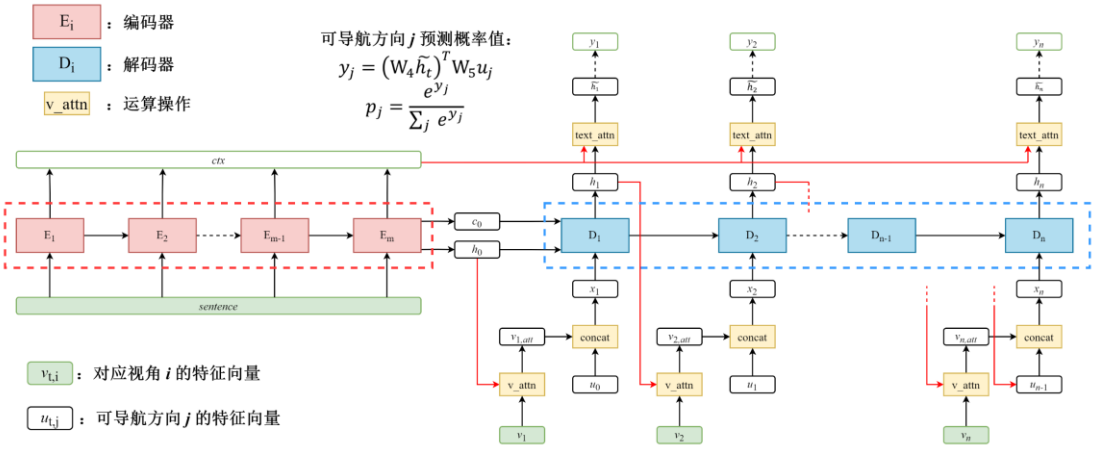

图 2-7 follower 模型结构图

编码器和解码器的基本结构都是长短期记忆单元（LSTM）。编码器部分负责处理自然语言指令（即给定导航路径的句子描述），对输入的导航语句中的每个单词进行编码。解码器部分负责处理视觉图像序列，每次输入到解码器中的是完整导航路径的视觉图像特征序列。编码器首先将自然语言指令编码成隐空间中的内容矩阵ctx即 context vector，然后解码器按顺序依次输入视觉图像特征序列中的当前帧，将视觉信息降维成隐空间中的隐状态h_t，每一时间步的隐状态h_t最后与编码器输出的内容矩阵ctx进行基于注意力机制的运算，得到注意力向量，然后注意力向量和当前时间步可选动作编码向量进行计算，最终得到当前时间步的预测动作。

2.3.3 Follower 模型中的注意力机制

注意力机制是深度学习中一种重要的数据处理方法，它可以在算力有限的情况下，将计算资源合理地分配给模型中更重要的部分。注意力机制的产生受人类视觉系统的启发，当人类在观察外界视觉环境或图片时，往往不会把视野范围内的所有视觉信息全部都仔细地扫描一遍，而是更倾向于根据自己感兴趣的东西有选择性地重点关注某些主要的部分。然后对这些主要的部分投入更多的注意力。在深度学习中，一般来说，模型越复杂，模型的参数越多，则模型的性能就越强大，模型所存储的信息量也越大，但信息量增大会导致计算算力和存储资源的巨

大开销，引入注意力机制后，可以帮助模型对输入数据的不同维度特征赋予不同的注意力权重，在众多的信息中抽取出当前任务最重要的关键信息，同时降低对其他信息的关注程度，以此缓解信息过载的问题，降低计算资源开销，并提高任务处理的效率和准确性。

在 follower 模型中，解码器部分需要根据输入的视觉图像序列输出每一时间步的预测动作，为了决定应该往哪个方向移动，agent 首先需要根据解码器计算的上一时间步的隐状态h_{t-1}和当前时间步的全景视觉特征表示v_t进行注意力计算，得到不同视角的注意力权重，如式（2-9）和式（2-10）所示：

$$a_{t,i} = (\mathrm{W}_1 h_{t-1} + b_1)^T (\mathrm{W}_2 v_{t,i} + b_2) \tag{2-9}$$

$$\alpha_{t,i} = \frac{e^{a_{t,i}}}{\sum_i e^{a_{t,i}}} \tag{2-10}$$

其中h_{t-1}是解码器上一时间步的隐状态，$v_{t,i}$是当前时间步输入到解码器中的全景视觉空间对应于视角i的特征向量。W_1、W_2是对应的参数矩阵，b_1、b_2是对应的偏置项。$\alpha_{t,i}$即为当前时间步计算出的不同视角的注意力权重。

得到不同视角的注意力权重后，将注意力权重与原来的$v_{t,i}$对应相乘，然后加和，如式（2-11）所示：

$$v_{t,att} = \sum_i \alpha_{t,i} v_{t,i} \tag{2-11}$$

其中$v_{t,att}$是当前时间步全景视觉空间所有视角特征向量计算出的注意力特征表示向量，$v_{t,att}$和当前时间步的可导航方向编码向量u_t进行拼接，得到拼接向量$x_t = [u_t, v_{t,att}]$，拼接向量再输入到解码器中。

x_t输入到解码器后，被解码器映射到隐空间中，得到当前时间步隐状态向量h_t，然后h_t和编码器输出的内容矩阵ctx进行注意力计算，得到对自然语言指令不同单词的注意力权重，再根据注意力权重将原来的内容矩阵ctx的不同单词进行加权求和，得到带权内容向量ctx_w，最后ctx_w和h_t拼接起来，再经过一个线性变换函数和 tanh 函数处理后，得到解码器当前时间步输出的最终隐状态向量$\widetilde{h_t}$，如式（2-12）和式（2-13）所示：

$$h'_t = [ctx_w, h_t] \tag{2-12}$$

$$\widetilde{h_t} = \tanh(\mathrm{W}_3 h'_t + b_3) \tag{2-13}$$

最后，使用向量对应元素乘积的方法计算出当前时间步每个可导航方向j的概率值，如式（2-14）和式（2-15）所示：

$$y_j = \left(\mathrm{W}_4 \widetilde{h_t}\right)^T \mathrm{W}_5 u_j \tag{2-14}$$

$$p_j = \frac{e^{y_j}}{\sum_j e^{y_j}} \tag{2-15}$$

其中概率值p_j越大，则表示 agent 在当前时间步最应该前往可导航方向j。然后，agent 就会按照指定导航方式（Teacher Forcing 或 Student Forcing）选择一个可导航方向j并朝着该方向前往对应的位置，特殊的，当 agent 认为自己已经到达目的地附近时，可以执行“STOP”动作编码向量u_0选择停止当前的导航任务。由于 follower 模型采用全景动作空间控制，所以 agent 可以直接无缝地将全景环境感知和动作转化为视觉运动控制，而不需要执行低级的动作如左转 30 度，右转 30 度。

2.4 本章小结

本章首先介绍了循环神经网络和长短期记忆网络的相关理论，然后介绍了低级运动控制方式的挑战与不足，并随即介绍了基于全景动作空间的高级运动控制方式，最后对本文参考的 follower 模型的具体结构和 follower 模型中的注意力机制进行了介绍。

第三章 Contrast-VisionVAE-Follower 模型

本章介绍具体的房间到房间（Room-to-Room，R2R）任务和基于 follower 模型改进后的 contrast-visionVAE-follower 模型。

3.1 任务描述

R2R 任务要求嵌入在 Matterport3D 模拟器中的 agent 能够通过理解给定的自然语言指令和环境视觉信息，从初始位置导航到目标位置。更加详细的定义是，在每一次单独的导航任务的开始，agent 会被给定一条自然语言指令$\bar{x} = \langle x_1, x_2, \dots, x_L \rangle$作为输入，其中$L$是自然语言指令的长度，$x_i$是自然语言指令中每个单词的词向量。同时，agent 还会通过在模拟器中进行一次观察获得初始 RGB 图像o_0，获得的初始 RGB 图像由 agent 的初始姿态决定，即$s_0 = \langle v_0, \psi_0, \theta_0 \rangle$，其中$s_0$是初始姿态，$v_0$是 3D 位置，$\psi_0$是航向角，$\theta_0$是仰角。agent 必须通过执行一系列连续的动作$\langle s_0, a_0, s_1, a_1, \dots, s_T, a_T \rangle$来改变自身的姿态或位置，其中每个被执行的动作$a_t$都会导致 agent 前往新的状态$s_{t+1} = \langle v_{t+1}, \psi_{t+1}, \theta_{t+1} \rangle$并进行新的观察生成新的 RGB 图像$o_{t+1}$。当 agent 执行特殊的“STOP”动作后，当前导航任务结束。如果执行的动作序列最后能让 agent 移动到距离目标位置v^*足够近的有效范围内，则认为该次导航任务成功，否则失败。

3.2 Contrast-VisionVAE-Follower 模型的设计

本论文改进的 contrast-visionVAE-follower 模型以 follower 模型[7]为基础，在原来的模型架构之上主要做出了以下改进：

（1）在视觉图像序列处理部分增加了一个视觉变分自编码器模块（visionVAE），用于学习视觉图像序列的概率分布；

（2）针对模型需要处理的自然语言指令和视觉图像序列这两种不同模态的数据，增加了一个语言和视觉信息跨模态对比学习模块，用于有效地学习自然语言指令和视觉图像序列跨模态信息的匹配关系。

改进后的 contrast-visionVAE-follower 模型结构如图 3-1 所示。

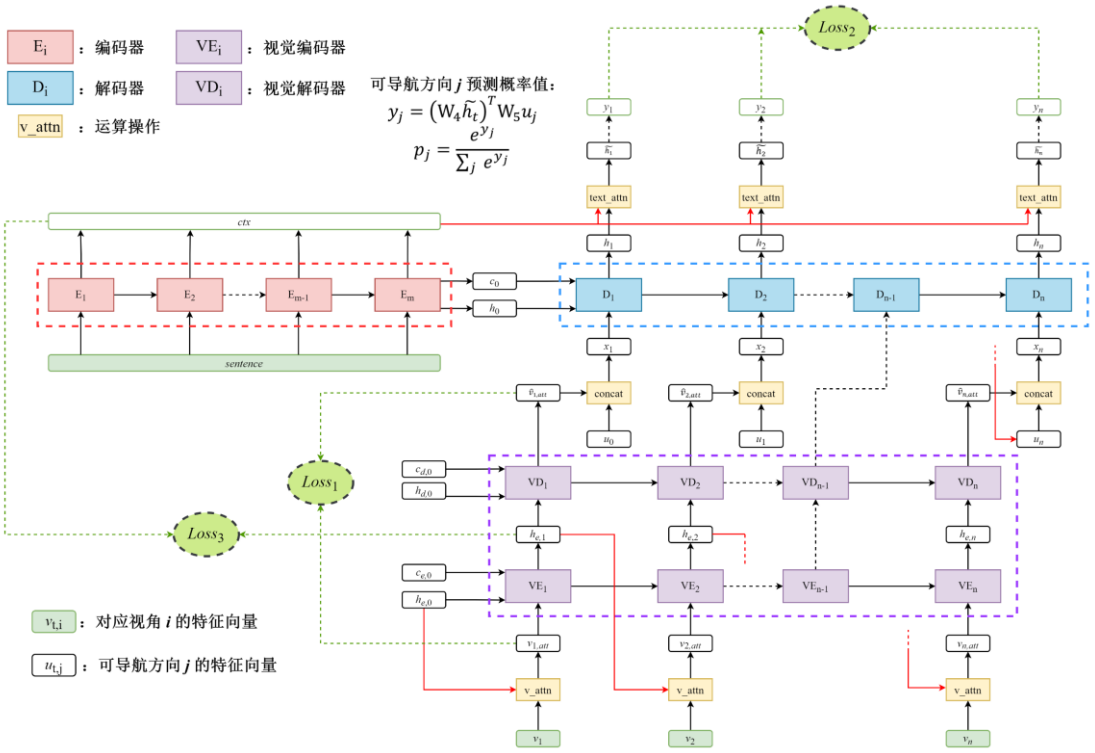

图 3-1　contrast-visionVAE-follower 模型结构图

3.2.1　变分自编码器

变分自编码器（Variational Auto-Encoder，VAE）[24]是一种无监督学习生成式模型，输入一组数据样本，模型可以输出一组与输入数据相似但不完全相同的新数据，使得生成的数据具有多样性。变分自编码器是在自编码器（Auto-Encoder，AE）的基础上发展的。

自编码器的模型结构由两部分组成，包含一个编码器和一个解码器，形成“encoder-decoder”结构。编码器的作用是将输入的数据进行编码降维，在降维的过程中，会形成“数据瓶颈”，使得模型可以学习到数据的重要维度特征。解码器的作用与编码器的作用相反，用于对已编码的隐空间向量进行解码，使隐空间向量从低维隐空间映射回原始数据维度空间中去，从而重构出原始数据。训练出的自编码器希望使得重构的数据与原来的数据误差越小越好。自编码器模型的结构如图 3-2 所示。

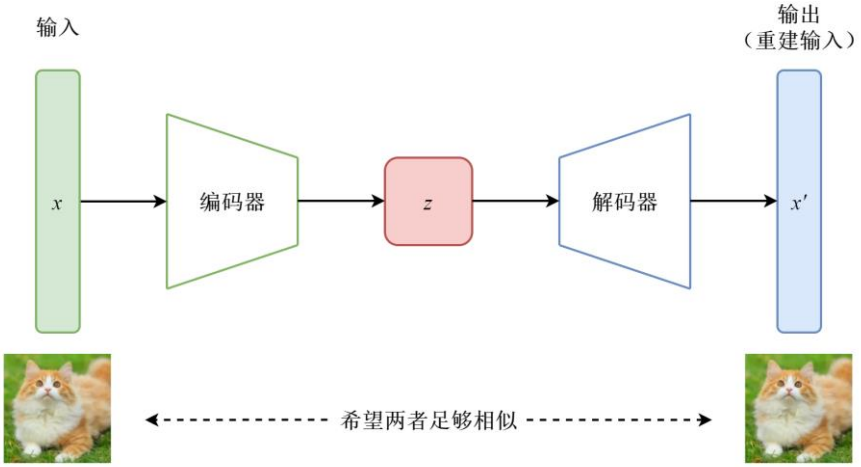

图 3-2　自编码器模型结构图

自编码器虽然可以重建输入数据，但是不能根据已有的数据生成新的数据，而变分自编码器则可以。变分自编码器与自编码器的最大不同是变分自编码器模型引入了概率分布思想，将原来确定的映射关系变成了一种服从一定概率分布的抽样过程。自编码器降维后的隐变量输出是一个具体的数值，而变分自编码器降维后的隐变量输出是一个概率分布，然后再从这个概率分布中进行采样，采样得到的隐变量再送入到解码器中进行解码，最后输出与原始数据相似但不完全相同的新数据，达到生成新数据的效果。变分自编码器模型的结构如图 3-3 所示。

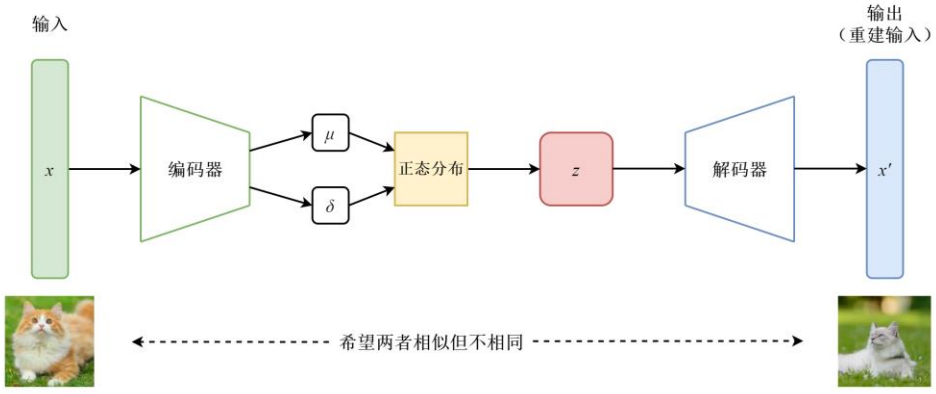

图 3-3　变分自编码器模型结构图

下面推导变分自编码器的目标函数。我们希望变分自编码器能够从数据集中取一个样本x输入到编码器中，编码器输出与当前的输入x对应的隐变量z的分布$p(z|x)$，从$p(z|x)$中进行采样得到对应的隐变量z，再输入到解码器中，解码出一个新的数据。原始数据x受编码后的隐变量z的影响，只要知道了隐变量z的分布$p(z)$就可以通过从隐变量分布中采样再进行解码操作得到输出值。即，我们希望

在给定x的情况下求出z的分布$p(z|x)$，根据贝叶斯定理，有式（3-1）：

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)} \tag{3-1}$$

从式（3-1）可知，想要求出$p(z \mid x)$，必须先求出分母中的$p(x)$，然而，真实数据的分布$p(x)$是无法获知的，因此我们无法根据上式求出$p(z \mid x)$。不过，虽然$p(z \mid x)$无法直接求解，但是我们可以引入一个新的分布$q(z \mid x)$，希望用分布$q(z \mid x)$去逼近分布$p(z \mid x)$，只要$q(z \mid x)$与$p(z \mid x)$足够的相似，那么我们就可以认为从$q(z \mid x)$中进行采样的操作等价于从$p(z \mid x)$中采样的操作。为了简单起见，一般认为假定的分布$q(z \mid x)$服从高维高斯分布，如式（3-2）所示：

$$q(z \mid x) \sim N(\mu_I, \sigma_I^2 I) \tag{3-2}$$

使用 KL 散度（Kullback-Leibler Divergence）衡量分布$q(z \mid x)$与分布$p(z \mid x)$之间的相似程度，如式（3-3）所示：

$$KL(q(z \mid x)||p(z \mid x)) = \int q(z \mid x)\log \frac{q(z \mid x)}{p(z \mid x)} dz \tag{3-3}$$

目标是最小化 KL 散度，为此，将 KL 散度展开，如式（3-4）所示：

$$\begin{aligned} KL(q(z \mid x) \parallel p(z \mid x)) &= \int q(z \mid x)\log \frac{q(z \mid x)}{p(z \mid x)} dz \\ &= \int q(z \mid x)\log \frac{q(z \mid x)}{\frac{p(x \mid z)p(z)}{p(x)}} dz \\ &= \int q(z \mid x)\log\, q(z \mid x)dz + \int q(z \mid x)\log\, p(x)dz \\ &\quad -\int q(z \mid x)\log\,[p(x \mid z)p(z)]dz \\ &= \log p(x) + \int q(z \mid x)\log\, q(z \mid x)dz \\ &\quad -\int q(z \mid x)\log\,[p(x \mid z)p(z)]dz \end{aligned} \tag{3-4}$$

我们的目标是最小化最后得到的展开式，注意到，原始数据的真实分布$p(x)$虽然不可知，但是是完全确定的，因此相当于一个常数，所以最小化展开式等价于最小化展开式的后两项，记作L_1，则对L_1做等价代换如式（3-5）所示：

$$\begin{aligned} L_1 &= \int q(z \mid x)\log\, q(z \mid x)dz - \int q(z \mid x)\log\,[p(x \mid z)p(z)]dz \\ &= \int q(z \mid x)\log\, q(z \mid x)dz \\ &\quad -\int q(z \mid x)\log\, p(x \mid z)dz - \int q(z \mid x)\log\, p(z)dz \\ &= \int q(z \mid x)\log \frac{q(z \mid x)}{p(z)} dz - \int q(z \mid x)\log\, p(x \mid z)dz \\ &= KL(q(z \mid x) \parallel p(z)) - E_{z\sim q(z|x)}[\log\, p(x \mid z)] \end{aligned} \tag{3-5}$$

最小化L_1等价于最大化$-L_1$，如式（3-6）所示：

$$\max\,(-L_1) = E_{z\sim q(z|x)}[\log\, p(x \mid z)] - KL(q(z \mid x) \parallel p(z)) \tag{3-6}$$

分析式（3-6）的意义。式子的第一项表示不断地在随机变量z上采样，使得被重构的样本中重构出的x的概率最大。式子的第二项表示希望我们最开始假设的分布$q(z \mid x)$与隐变量分布$p(z)$越相似越好（即它们之间的 KL 散度越小越好）。

一般假设分布$p(z)$服从高维标准高斯分布，即$p(z)\sim N(0,I)$，其中z的每一维之间都是相互独立的。

为了使用梯度下降算法进行训练，我们应该最小化目标函数，如式（3-7）所示：

$$min\ L_1 = -E_{z\sim q(z|x)}[\log p(x \mid z)] + KL(q(z \mid x) \parallel p(z)) \qquad (3-7)$$

上式即为变分自编码器在训练过程中需要优化的目标函数，目标函数越小越好。根据变分自编码器的目标函数，我们希望从分布$N(\mu_I, \sigma_I^2 I)$中采样出一个z，然后重构输入x，计算损失。然而，从分布中进行采样的这个操作是不可以计算导数的。为了使得模型可以进行梯度回传，需要使用如下操作进行代替：首先从高维标准高斯分布$N(0,I)$中采样一个随机噪声向量ε，然后将计算得到的分布$N(\mu_I, \sigma_I^2 I)$的均值和方差和ε进行运算，如式（3-8）所示：

$$z = \mu + \varepsilon \times \sigma \qquad (3-8)$$

通过这样的称为“重参数化”的操作，就可以顺利地进行求导操作，使得模型可以正常地梯度回传。

在实际编写模型代码时，对计算 KL 散度的过程使用式（3-9）的等价形式以便于计算机进行数值计算和梯度回传：

$$KL\big(q(z \mid x) \parallel p(z)\big) = \frac{1}{2}\sum_{j=1}^{J}[\mu_j^2 + \sigma_j^2 - log\ (\sigma_j^2) - 1] \qquad (3-9)$$

式（3-9）即为在实际代码中计算 KL 散度的计算式，其中μ_j为第j维的均值，σ_j为第j维的方差。

3.2.2 视觉变分自编码器模块

在 contrast-visionVAE-follower 模型中，增加的视觉变分自编码器模块（visionVAE）用于学习输入数据样本的视觉图像序列的概率分布，先编码真实视觉图像特征表示，再从视觉变分自编码器模块的编码隐空间概率分布中解码出相似但不完全相同的新视觉图像特征表示，使得在有限的训练数据情况下，通过视觉信息重建，增加训练时视觉图像序列信息的多样性，提高模型在未见过的视觉环境中的泛化性能。visionVAE 模块在模型中的结构如图 3-4 所示。

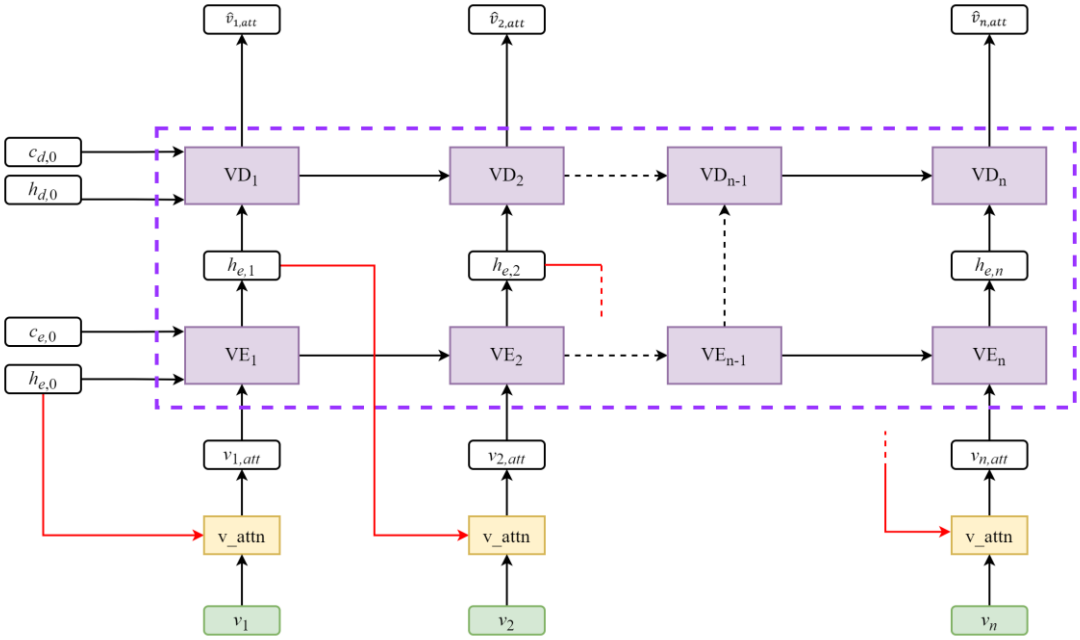

图 3-4　visionVAE 模块结构图

visionVAE 模块的视觉编码器部分采用的注意力机制与原来的 follower 模型中的解码器部分使用的注意力机制相同，根据上一时间步编码的隐向量$h_{e,t-1}$和当前时间步 agent 相机可观察到的周围所有视角图像特征v_t进行注意力计算，得到不同视角的注意力权重，如式（3-10）和式（3-11）所示：

$$a_{t,i} = \left(\mathrm{W}_1 h_{e,t-1} + b_1\right)^T \left(\mathrm{W}_2 v_{t,i} + b_2\right) \tag{3-10}$$

$$\alpha_{t,i} = \frac{e^{\alpha_{t,i}}}{\sum_i e^{\alpha_{t,i}}} \tag{3-11}$$

其中$h_{e,t-1}$是视觉编码器上一时间步t编码的隐向量，$v_{t,i}$是当前时间步输入到视觉编码器中的全景视觉空间所有视角的特征向量。W_1和W_2是对应的参数矩阵，b_1、b_2是对应的偏置项，$\alpha_{t,i}$即为当前时间步计算出的不同视角的注意力权重。

得到不同视角的注意力权重后，将注意力权重与原来的$v_{t,i}$对应相乘，然后加和，如式（3-12）所示：

$$v_{t,att} = \sum_i \alpha_{t,i} v_{t,i} \tag{3-12}$$

其中$v_{t,att}$是当前时间步全景视觉空间所有视角特征向量计算出的注意力特征表示向量，即 visionVAE 模块的原始输入。随后视觉编码器编码注意力特征表示向量$v_{t,att}$，输出当前时间步隐向量$h_{e,t}$（$h_{e,t}$除了用于视觉解码器的解码过程外，还将用于跨模态对比学习模块中），然后使用当前时间步隐状态$h_{e,t}$计算出当前数据样本对应的均值和对数方差，如式（3-13）和式（3-14）所示：

$$\mu_t = \mathrm{W}_3 h_{e,t} + b_3 \quad (3-13)$$

$$\log(\sigma_t^2) = \mathrm{W}_4 h_{e,t} + b_4 \quad (3-14)$$

之后为了使得梯度可以反向传播，采用上述所提到的“重参数化”策略，从高维标准高斯分布$N(0,I)$中随机采样一个噪声向量ε_t，计算出输入到视觉解码器中的隐向量$\hat{h}_{e,t}$，如下式所示：

$$\hat{h}_{e,t} = \mu_t + \varepsilon_t \times \sigma_t^2 \quad (3-15)$$

采样隐向量$\hat{h}_{e,t}$输入到视觉解码器中，经过解码重建出一个新的视觉特征表示向量$\hat{v}_{t,att}$。当 visionVAE 模块训练得足够好的时候，可以使得重建出来的$\hat{v}_{t,att}$与原始输入的$v_{t,att}$相似但不相同。因此，每一时间步 visionVAE 模块的损失函数$Loss_{t,1}$包含两部分，即视觉特征表示的重建损失和隐变量分布的相似度损失，如式（3-16）所示：

$$Loss_{t,1} = \frac{1}{d}\left\|v_{t,att} - \hat{v}_{t,att}\right\|_2^2 + \frac{1}{2}\sum_{j=1}^{J}\left(\mu_{t,j}^2 + \sigma_{t,j}^2 - \log\left(\sigma_{t,j}^2\right) - 1\right) \quad (3-16)$$

其中d是视觉特征表示向量$v_{t,att}$的维度数，$\mu_{t,j}$是第j维的均值，$\sigma_{t,j}$是第j维的方差。$Loss_{t,1}$的值越小，则视觉变分自编码器模块训练得越好。

与原始的 follower 模型中的解码器部分相同，重建出的$\hat{v}_{t,att}$和当前时间步的可导航方向编码向量u_j进行拼接，得到拼接向量$x_t = [u_t, \hat{v}_{t,att}]$，拼接向量再输入到解码器中，进行高级动作的预测计算。最后计算每一时间步的预测动作和真实动作标签之间的交叉熵损失$Loss_{t,2}$。在计算$Loss_{t,2}$时，对于不同时间步的动作预测损失，乘上一个与时间步有关的递增超参数γ_t。因为随着时间步的增加，agent 执行的预测动作的偏差将逐渐累加，越靠后的动作如果预测效果不准，则将使得 agent 越偏离正确的目的地，因此超参数γ_t可以用于惩罚 agent 在训练过程中的预测不准确的动作。超参数γ_t的计算方式与乘上超参数γ_t后的最终预测动作与真实动作标签的交叉熵损失如式（3-17）和式（3-18）所示：

$$\gamma_t = \sqrt{\frac{l_{episode} + \beta t}{l_{episode}}} \quad (3-17)$$

$$Loss_{t,2} = \gamma_t Loss'_{t,2} \quad (3-18)$$

其中$l_{episode}$是模型在进行动作预测时所允许的最大时间步数，超参数$\beta = 0.5$，$Loss'_{t,2}$是原始的交叉熵损失，$Loss_{t,2}$是最终的交叉熵损失。

3.2.3 对比学习

对比学习（Contrastive Learning）是一种自监督学习模型，自监督学习是无监督学习范式的一类。与常规的监督学习不同，自监督学习最大的特点是不需要人工进行标注的真实标签信息，直接利用输入的原始数据作为监督信息，训练模

型提取数据的有效特征表示，训练出的模型再根据具体的下游任务进行微调。

对比学习的一般方式是将原始数据样本通过某种处理手段分成正例样本和负例样本，然后模型将原始样本、正例样本和负例样本通过相同的编码方式映射到相同的隐空间中进行对比，拉近原始数据和正样本的距离，同时拉远原始数据和负样本的距离，从而使模型学到原始数据之间的良好特征表示。因此，对比学习的一个关键问题就是如何合理的构造出正样本和负样本对。

对比学习的一般范式是将输入的数据构造成正例样本和负例样本，学习一个编码器f，将正例样本和负例样本输入到编码器中编码成隐空间的特征表示$f(x)$，使得编码器f满足式（3-19）：

$$\mathrm{score}\big(f(x), f(x^+)\big) \gg \mathrm{score}\big(f(x), f(x^-)\big) \tag{3-19}$$

其中x是原始数据，x^+是和x相似的正例样本，x^-是和x不相似的负例样本，$score(\cdot)$是一个用于衡量两个数据之间相似度的度量函数或指标。一个比较常用的度量函数是向量内积，如果使用向量内积计算两个数据之间的相似度，则对比学习模型最后要优化的损失函数如式（3-20）所示：

$$L_N = -\mathbb{E}_X\left[\log \frac{\mathrm{e}^{f(x)^T f(x^+)}}{\mathrm{e}^{f(x)^T f(x^+)} + \sum_{j=1}^{N-1} \mathrm{e}^{f(x)^T f\left(x_j^-\right)}}\right] \tag{3-20}$$

从上式可知，一个样本x对应有一个正例样本x^+和$N-1$个负例样本x^-，从形式上看，该损失函数相当于是常规的交叉熵函数，等价于在做一个N分类的任务，正例样本是正确的类别，而负例样本则是错误的类别。上式所示的对比学习损失函数一般在相关文献中被称为$InfoNCE$损失。

3.2.4 跨模态对比学习模块

在 contrast-visionVAE-follower 模型中，增加的跨模态对比学习模块用于学习语言和视觉跨模态信息的匹配关系，拉近相似跨模态信息在隐空间中的距离，拉远不相似跨模态信息在隐空间中的距离，使模型能更有效地编码语言信息和视觉信息。该模块主要参考了对比学习中一个有重大影响力的工作，即由何恺明等人[25]提出的 MoCo 方法。

MoCo 方法的主要贡献是提出了两个核心的操作，分别是样本存储队列和动量更新操作。正如前面提到，对比学习模型的一个关键问题是如何构造正例样本和负例样本，同时，当参与对比学习的负例样本的数量增多时，能增加模型训练的难度，提升模型对特征表示的有效提取能力。增加负例样本个数的典型方法如图 3-5 所示。

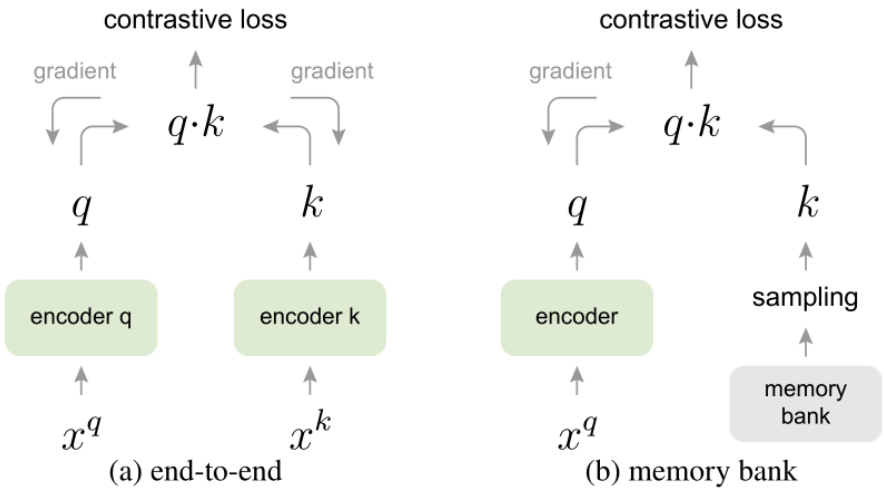

图 3-5　对比学习中增加负例样本的两种典型方法[25]

其中 end-to-end 方法是只要数据集有多少原始数据就构造出同样数量的负例个数，每一轮输入数据到模型中的时候，输入的负例个数都是整个数据集的数据个数。虽然，end-to-end 方法显著地增大了负例样本的个数，但是一般情况下数据集的容量是非常大的，而显卡的容量也有上限，因此采用 end-to-end 方法的模型的参数量和显存占用量也非常巨大，无法在计算机上进行训练。memory bank 方法则是预先使用一个 memory bank 结构存储下整个数据集中的所有数据编码后的特征表示，然后在每一次训练迭代的时候则是随机地从这个 memory bank 中抽取出一个批次的数据，然后再将数据输入到编码器中进行梯度更新。在这种方法中，虽然每次只是从 memory bank 中抽取一部分的数据，但是可以认为在经过多次的训练迭代后对每个小批次数据的更新近似于对整个数据集特征表示的更新。然而，memory bank 方法存在一个问题，就是预先保存数据集中的所有数据的特征表示仍然是非常占用 gpu 显存的，这会限制模型其他部分的结构，影响模型性能。

MoCo 方法如图 3-6 所示。该方法认为可以将对比学习看做是一个训练编码器执行在动态变化的字典中查找匹配信息的过程，提出了将使用 memory bank 的方法改进为使用一个动态变化的队列，这个队列的作用跟 memory bank 类似，也是用于保存数据集中的特征表示，不同点是这个队列里存储的数据特征是动态更新的，在每一次训练迭代时都会对队列进行入队操作，将当前编码器编码的一个批次的数据特征存入队列中，同时删除队列中保存时间最久的一个批次的数据特征，整体上来看每个训练迭代，队列中保存的数据特征的总数是不变的，并且会随着训练过程的进行更新队列中的数据特征。这样，我们就可以灵活的地根据需要设置不同容量大小的队列，避免了占用 gpu 显存较大的问题。

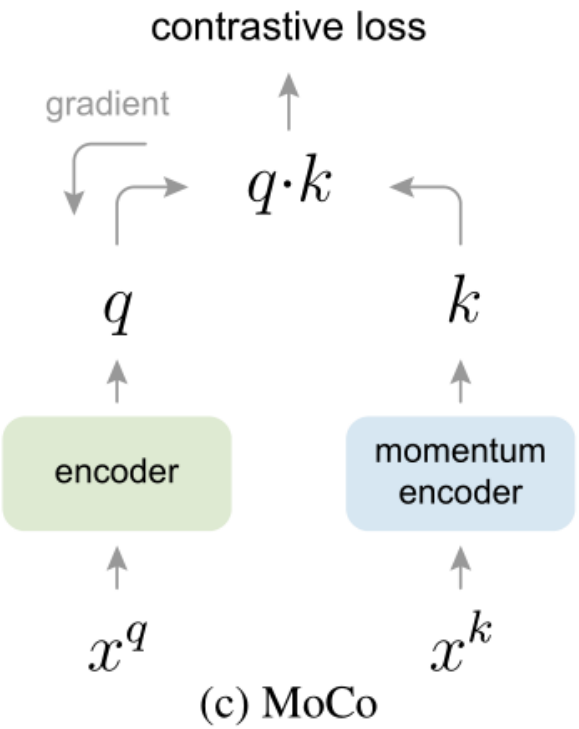

图 3-6　MoCo 方法的对比学习结构图[25]

在跨模态对比学习模块中，由视觉变分自编码器中的视觉编码器编码的每一时间步的隐向量$h_{e,t}$都将被拼接起来形成隐向量序列。由于模型在实际训练时是按一个批次输入数据样本的，因此在一个批次的数据样本中，有的样本在预测动作序列时可能会提前执行“STOP”动作，因此一个批次中不同样本输出的预测动作序列长度是不一致的，因此为了对齐，将长度小于所允许最大时间步数$l_{episode}$的隐向量序列末尾统一用最后时间步的$h_{e,t}$进行填充，最后合并成一个隐向量矩阵H。

至此，当一个批次的数据样本完整地在 contrast-visionVAE-follower 模型中被处理完成后，可以得到内容矩阵ctx和隐向量矩阵H，由于矩阵ctx的句子序列维度和矩阵H的动作序列维度长度不一致，因此需要先进行对齐处理，转化为维度一致的方阵，如式（3-21）和式（3-22）所示：

$$\mathrm{Q}' = \mathrm{ctx}^T \mathrm{ctx} \tag{3-21}$$

$$\mathrm{K}' = \mathrm{H}^T \mathrm{H} \tag{3-22}$$

然后对方阵Q′和方阵K′使用 softmax 函数对每一行i进行归一化处理，如式（3-23）和式（3-24）所示：

$$\mathrm{W}_{\mathrm{Q},i} = \mathrm{softmax}(\mathrm{Q}'_i) \tag{3-23}$$

$$\mathrm{W}_{\mathrm{K},i} = \mathrm{softmax}(\mathrm{K}'_i) \tag{3-24}$$

之后将方阵$\mathrm{W_Q}$和方阵$\mathrm{W_K}$与各自对应的$\mathrm{Q'}$、$\mathrm{K'}$进行每一行元素对应乘积然后求和运算，如式（3-25）和式（3-26）所示：

$$q_i = \sum_j \mathrm{W_{Q,}}_{i,j}\mathrm{Q'}_{i,j} \tag{3-25}$$

$$k_i = \sum_j \mathrm{W_{K,}}_{i,j}\mathrm{K'}_{i,j} \tag{3-26}$$

向量q相当于对比学习模型中的原始样本，向量k相当于正例样本，同时还有一个用于存储负例样本的队列Q，长度为K，K即为队列中存储的负例样本个数。注意，q、k和Q中的隐空间维度要使用$L2$范数进行归一化处理。

对q和k进行$L2$范数归一化处理后，计算原始样本和正例样本的相似度$score_{pos} = q^T k$，然后计算原始样本和负例样本队列的相似度$score_{neg} = q^T \mathrm{Q}$，之后将$score_{pos}$和$score_{neg}$拼接起来，得到正例相似度在索引值为 0 处的相似度向量$score_{total} = [score_{pos}, score_{neg}]$。

由于正例相似度全在向量中的第一位，因此对比学习希望训练模型使得向量中第一位的相似度值最大，其他位的相似度越小，因此可以将对比学习损失转换成一个$K+1$分类任务的分类交叉熵损失$Loss_3$，如式（3-27）所示：

$$Loss_3 = -\sum_{i=1}^{K+1} y_{s,i} \log\left(score_{total,i}\right) \tag{3-27}$$

其中$y_{s,i}$是真实的第i类的概率值。最终整个 contrast-visionVAE-follower 模型的损失函数$Loss$由$Loss_1$、$Loss_2$和$Loss_3$三部分组成，如式（3-28）所示：

$$Loss = Loss_1 + Loss_2 + Loss_3 \tag{3-28}$$

在训练过程中，希望$Loss$的值越小越好。

3.3 本章小结

本章首先对视觉语言导航中的房间到房间任务进行了具体的描述，然后提出了对 follower 模型的改进，介绍了改进后的 contrast-visionVAE-follower 模型的具体结构，给出了对模型进行改进的相关理论依据即变分自编码器和对比学习的介绍，并具体介绍了改进模型中增加的视觉变分自编码器模块和跨模态对比学习模块。

第四章　实验结果与分析

4.1　R2R 数据集

本实验所使用的数据集为 R2R 数据集。R2R 数据集由 10800 个全景视图组成，这些全景视图由 90 个达到完整建筑物规模级别的真实场景的 194400 张 RGB-D 图像构成。这些全景视点在每个场景的可以行走的楼层和平面上较为均匀地分布，视点和视点之间的平均距离为 2.25m。每个全景视图都是从一个 3D 位置大致为一个站立的人的高度观察到的 18 张 RGB-D 图像构成。每张图像都附带有一组精确的 6 自由度摄像机姿态的注释。这些图像捕获了除极点以外的整个可视球体的视觉信息。

总体来说，在视觉多样性方面，R2R 数据集中所包含的场景囊括了大部分的建筑物风格，包括大大小小和复杂程度不同的房屋、公寓、酒店、办公室和教堂等。因此，R2R 数据集具有巨大的视觉多样性，可以对相关的计算机视觉任务构成真正的挑战。

4.2　数据集划分

R2R 数据集由从 Matterport3D 模拟器导航图中采样的 7189 条路径组成，每条路径由 5 到 7 个离散视点组成，所有路径的平均路径长度为 10m。每条路径配有三条符合人类语言的指令句子，共有约 21500 条，每条指令平均有 29 个单词。R2R 数据集被划分为训练集、验证集和测试集三部分，其中验证集又被分为可见验证集和不可见验证集两部分，可见验证集的数据样本在训练集中出现过，而不可见验证集的数据样本则是在训练集中从未出现过的。所有测试集的数据样本都从未在训练集和验证集中出现过。训练集和可见验证集包含 61 个场景，训练集有 14025 条指令数据，可见验证集有 1020 条指令数据。不可见验证集包含 11 个场景和 2349 条指令，测试集包含 18 个场景和 4173 条指令。本实验只用到训练集和验证集两部分。

4.3　Matterport3D 模拟器

本实验需要在 Matterport3D 模拟器中进行训练。Matterport3D 模拟器是一种

新的大规模视觉强化学习模拟环境，可以被应用于基于 R2R 数据集的 agent 的科学研究与开发工作[26]。

Matterport3D 模拟器是在 OpenGL 上使用 C++高级程序语言编写的。不过，Matterport3D 模拟器除了提供基于 C++语言的 API（应用程序编程接口）之外，还提供基于 Python 语言的 API，这使得该模拟器可以很轻松地与常见的深度学习框架如 Caffe 和 TensorFlow 或强化学习平台如如 ParlAI 和 OpenAI Gym 配合使用。Matterport3D 模拟器提供了多种常规的相机配置参数和选项，例如图像分辨率、相机画幅高度、相机画幅宽度和视野参数等。除了模拟器本身之外，研究人员还开发了一个基于 WebGL 浏览器的可视化库，可以用于使用 Amazon Mechanical Turk 来进行人工地收集导航轨迹的文本注释，该可视化库提供给有需要的研究人员。

4.3.1 环境观察

为了使得模型的 agent 可以与 Matterport3D 模拟器进行交互，采用使 agent 的姿态与模拟器中的全景视点保持一致的方式，使得嵌入在 Matterport3D 模拟器中的 agent 可以在整个场景环境中进行虚拟的移动。

agent 在 Matterport3D 模拟器中的姿态由 3 个参数完全定义，即，在模拟器中的 3D 位置$v \in V$，相机航向角$\psi \in [0,2\pi)$和相机仰角$\theta \in \left[-\frac{\pi}{2},\frac{\pi}{2}\right]$。其中$V$是与模拟器场景中的全景视点相关联的 3D 点集。在某次具体的导航任务中，按照从导航开始到导航结束的时间顺序划分，在每个时间步t，模拟器都会输出与 agent 的第一人称摄像机视图相对应的 RGB 图像观察o_t。这些 RGB 图像观察是在模拟器场景中从每个视点的预计算立方体映射图像根据透视投影原理生成的。

4.3.2 动作空间

确定 agent 在模拟器中进行交互时的执行动作与移动位置之间的明确关系的关键是确定依赖于状态的动作空间，同时还要根据实际情况防止 agent 通过墙壁和地板这些不可穿透的障碍直接进行移动。因此，在每一时间步t，模拟器除了会输出上述所提到的 RGB 图像观察o_t外，还会输出一组下一时间步可到达的视点$W_{t+1} \subseteq V$。agent 通过选择下一时间步的新的视点$v_{t+1} \in W_{t+1}$来与模拟器进行交互，并指定 agent 下一时间步调整的相机航向角$\Delta\psi_{t+1}$和相机仰角$\Delta\theta_{t+1}$，agent 就是这样连续地在模拟器中进行移动，直到当前的导航任务结束。因此，agent 在模拟器中每一时间步的移动和位置

是完全确定的，只与上一时间步的动作有关。

为了确定下一时间步可到达的视点W_{t+1}，在模拟器中每个场景都基于全景视点构建出一个加权无向图$G = \langle V, E \rangle$，如果在加权无向图中两个视点V_i和V_j之间可以互相连通，即 agent 可以从视点V_i畅通无阻地移动到视点V_j，则称视点V_i和V_j之间存在一条边E，边的权重表示视点V_i和V_j之间的直线路径距离。

给定导航图G，下一步可到达的视点集如式（4-1）所示：

$$W_{t+1} = \{v_t\} \cup \{v_i \in V \mid \langle v_t, v_i \rangle \in E \wedge v_i \in P_t\} \qquad (4-1)$$

其中v_t是 agent 当前时间步t所在的视点，P_t是当前时间步由 agent 第一人称摄像机的视锥的左、右可视范围所包围的空间区域。该表示式的意义是，agent 下一时间步的移动可以有两种选择：

（1）留在当前视点v_t原地不动；

（2）在 agent 摄像机可以观察到的空间区域内，选择移动到与当前视点v_t存在连接边的其他视点v_j，且$v_j \neq v_t$。

即，在当前时间步，只要下一个要前往的位置在 agent 的当前视野内且在导航图中存在连接边，或 agent 可以通过上下左右移动摄像头看到，那么 agent 就可以畅通无阻地沿着导航图中的边移动到该位置，或者，agent 可以选择保持在当前视点不动，只是简单地移动相机的航向角或仰角。

图 4-1 显示了导航图的典型示例。根据对所有场景的加权无向图进行统计，每个加权无向图平均包含 117 个视点，每个视点的平均顶点个数为 4.1。总体来说，虽然真实环境中可以无限连续运动的场景被离散化成了一个个加权无向图，但是在大多数的高级导航任务中，离散化的视点位置和动作不会对研究的进行构成重大误差和限制，因为即使是在现实环境中的一个真正的机器人，当它在当前时间步从搭载的摄像机中观察到了周围的图像信息，它也没有必要去对所有真实空间中所有连续分布的可以到达的高级层次目的目标全部一一分析，因为这无论是在时间消耗上还是在计算机算力上都是消耗巨大的。因此，实际上，在理论分析和模拟仿真中，即使 agent 是在支持连续运动的常见 3D 模拟器中进行移动，在实践中也通常使用离散化的动作空间。

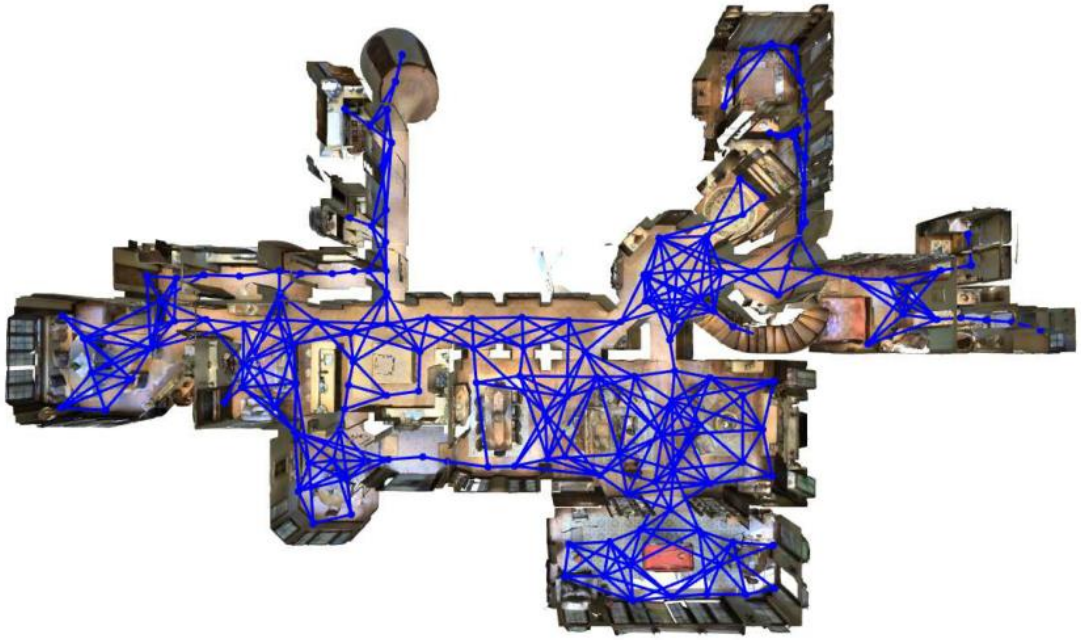

图 4-1　Matterport3D 模拟器中一个建筑场景的部分楼层的导航图示例[1]

Matterport3D 模拟器不会对 agent 在执行特定任务时的目标、奖励函数或任何其他的额外信息进行明确定义或限制，因为这些内容是根据使用者所执行的特定任务和特定数据集所决定的，因此灵活性和自由度较高。

4.4　实验环境配置

4.4.1　服务器环境

实验室的服务器为基于 Linux 内核的 Ubuntu 操作系统，Ubuntu 系统的版本号为 18.04。服务器上装有两张 NVIDIA GeForce RTX 3090 显卡。本实验使用一张 3090 显卡即可。

4.4.2　下载 R2R 数据集与 Matterport3D 模拟器文件

下载实验所需要的 R2R 数据集。首先进入 R2R 数据集的官方网站，签署一份使用条例，然后将签署的使用条例发送到官方指定的邮箱 matterport3d@googlegroups.com，很快就会收到官方的回复，在回复中就会附带一个用于下载 R2R 数据集的 python 文件，在服务器命令行中执行该 python 文件即可下载 R2R 数据集。完整的 R2R 数据集大小约为 1.3T，由于本实验只需使用 Matterport3D 模拟器的仿真功能，所以只需要下载 R2R 数据集中的

matterport_skybox_images 和 undistorted_camera_parameters 这两部分的数据。在服务器命令行使用 python2.7 版本执行如下指令：

python download_mp.py -o /data/share/matterport3d/mP3Ddata --type matterport_skybox_images undistorted_camera_parameters

即可开始下载 matterport_skybox_images 和 undistorted_camera_parameters 的数据。其中/data/share/matterport3d/mP3Ddata 为保存下载数据的文件夹路径。下载完成的数据大小总共约为 20G，注意需要将下载的所有数据中的压缩包进行解压，这样才能保证后续设置环境变量将 Matterport3D 模拟器和数据集关联起来时不会出错。

然后从 github 网站上下载官方提供的 Matterport3D 模拟器的完整文件。使用 git 命令将 github 网站上的 Matterport3D 模拟器文件复制到本地服务器的对应文件夹中。在服务器命令行中输入以下指令即可：

Make sure to clone with --recursive

git clone --recursive https://github.com/peteanderson80/Matterport3DSimulator.git

cd Matterport3Dsimulator

执行指令后，下载的完整的 Matterport3D 模拟器文件就已经保存在服务器对应的文件夹中。之后需要设置环境变量，将模拟器和 R2R 数据集关联起来。在服务器命令行输入如下指令将环境变量的值设置为之前已经解压的数据集的文件夹位置：

export MATTERPORT_DATA_DIR=<PATH>

注意<PATH>的值是包含所有 90 个建筑对应数据文件的上一级文件目录的完整绝对路径，而不是相对路径或符号链接。本服务器上存放数据集的完整绝对路径为/data/share/matterport3d/mP3Ddata/data/v1/scans，其中文件夹 scans 包含所有 90 个建筑的对应文件夹。因此应该输入如下确定的环境变量设置指令：

export MATTERPORT_DATA_DIR=/data/share/matterport3d/mP3Ddata/data/v1/scans

4.4.3 安装模拟器依赖项

模拟器的发布者建议使用 docker 文件来安装模拟器。docker 是一个开源的软件集装箱化应用引擎，可以让开发者在构建自己的各种不同的应用程序时，将他们的应用和依赖文件全部打包进一个可以移植的镜像容器中，然后就可以很轻松地将对应的镜像容器发布到任意平台如使用 Linux 或 Windows 等操作系统的机器上。因此，使用 docker 文件安装 Matterport3D 模拟器可以将繁琐复杂的代

码集合封装在一起，安装起来会非常方便，而且安装 Matterport3D 模拟器的环境可以与 Linux 服务器的环境和其他应用的环境隔离开，避免了由于安装模拟器的依赖文件而破坏掉服务器上原来已有的环境配置，造成无法预料的安全隐患。当然，Matterport3D 模拟器也可以在不使用 docker 文件的情况下进行安装，但是不适用 docker 的安装方式会比较繁琐，而且需要使用者谨慎地考虑各个所需依赖文件的关系，安装起来可能会更加困难。本实验采用 docker 文件的方式安装 Matterport3D 模拟器。

在使用 docker 安装 Matterport3D 模拟器之前，服务器上的配置需要达到如下要求：

（1）Nvidia GPU with driver >= 396.37；

（2）安装 docker；

（3）安装 nvidia-docker2.0。

第一步要求服务器已经满足，第二步需要安装 docker。打开 docker 官网，找到对应于不同服务器版本的 docker 安装包如图 4-2 所示。

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	s390x
CentOS	✓	✓		
Debian	✓	✓	✓	
Fedora	✓	✓		
Raspbian			✓	
RHEL				✓
SLES				✓
Ubuntu	✓	✓	✓	✓
Binaries	✓	✓	✓	

图 4-2　不同服务器版本的 docker 安装包

本服务器是 Ubuntu 系统，因此选择对应的 Ubuntu x86_64/amd64 版本。首先使用以下指令卸载旧版本的 docker：

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

在新的服务器上第一次安装 docker 引擎之前，需要先设置 docker 存储库，便于之后可以从存储库中安装和更新 docker。更新 apt 软件包索引和安装包以允许 apt 命令通过 HTTPS 网络使用存储库。设置 docker 存储库的指令如下：

```
sudo apt-get update
```

```
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

然后添加 docker 的官方 GPG 秘钥：

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

最后使用以下指令来设置稳定的存储库：

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

存储库设置完成后，更新 apt 软件包索引，并安装最新版本的安装 docker 引擎。指令如下：

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

至此，docker 引擎安装完成。

最后一步安装 nvidia-docker2.0。首先使用如下指令设置 docker：

```
curl https://get.docker.com | sh \
&& sudo systemctl --now enable docker
```

然后使用如下指令设置软件包存储库和 GPG 秘钥：

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
      && curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
      && curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.list | \
            sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
            sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

更新软件包列表后，安装 nvidia-docker2 软件包和依赖项：

```
sudo apt-get update
sudo apt-get install -y nvidia-docker2
```

设置默认运行时间后，重新启动 docker 守护程序以完成安装：

```
sudo systemctl restart docker
```

至此，nvidia-docker2.0 安装完成。

4.4.4 安装 Matterport3D 模拟器

使用 docker 文件安装 Matterport3D 模拟器。进入到之前下载好的 Matterport3D 模拟器的文件夹中，文件夹中有一个 docker 文件用于构建模拟器对应的 docker 镜像，在服务器的命令行中输入以下指令：

```
docker build -t mattersim:9.2-devel-ubuntu18.04 .
```

模拟器的 docker 镜像会自动地从远程拉取到本地服务器，等待一段时间后即可安装完成。然后运行该 docker 容器，挂载 Matterport3D 模拟器文件和 R2R 数据集：

```
nvidia-docker run -it --mount type=bind,source=$MATTERPORT_DATA_DIR,target=/root/mount/Matterport3DSimulator/data/v1/scans --volume `pwd`:/root/mount/Matterport3DSimulator mattersim:9.2-devel-ubuntu18.04
```

成功进入 docker 容器后，保持自己的命令行路径仍是 Matterport3D 模拟器文件夹所在的位置不动，在命令行中依次输入以下指令：

```
cd /root/mount/Matterport3DSimulator
mkdir build && cd build
cmake -DEGL_RENDERING=ON ..
make
cd ../
```

等待一段时间后，Matterport3D 模拟器和环境就已经全部搭建好了。为了在模型训练的时候能够加快数据的加载速度并减少内存的使用，需要通过缩小尺寸并将所有立方体面合并为一幅图像来预处理 R2R 数据集中的 matterport_skybox_images 文件下的图像。在 docker 容器中运行以下指令：

```
./scripts/downsize_skybox.py
```

完成后，数据集中的 matterport_skybox_images 子目录将包含文件名格式为 <PANO_ID>_skybox_small.jpg 的图像文件。

4.5 实验设置与结果分析

在 Matterport3D 模拟器中对基础 follower 模型和改进后的 contrast-visionVAE-follower 模型分别进行多组对照实验，分析实验结果和改进模型的性能。

4.5.1 模型训练参数设置

在 follower 模型和 contrast-visionVAE-follower 模型中，一些训练参数的设置完全相同，如下表所示：

表 4-1 模型相同训练参数设置一览表

参数名称	数值大小	参数解释
batch size	100	每一次输入到模型中的数据样本数量
max episode len	80	在将一个批次数据中的句子根据词表的索引值构造成矩阵时的句子长度的维度大小
max input len	10	在模型根据输入的视觉图像序列按顺序输出预测动作时，允许的最大输出次数
word embedding size	300	句子单词编码成向量的维度大小
action embedding size	2048+128	高级动作编码成向量的维度大小，其中 2048 维用于视觉特征表示部分，128 维用于 agent 姿态编码表示部分
hidden size	512	模型中隐藏层向量的维度大小
dropout ratio	0.5	神经网络中在训练时随机丢弃神经元的概率值
learning rate	0.0001	模型的学习率
weight decay	0.0005	模型优化器的权重衰减率
feature size	2048+128	可导航方向视觉信息编码成向量的维度大小，其中 2048 维用于视觉特征表示部分，128 维用于 agent 姿态编码表示部分
iterations	20000	模型输入一个批次的数据样本为一次迭代，总共训练迭代 20000 次

follower 模型和 contrast-visionVAE-follower 模型在训练过程中都使用 Adam 优化器更新模型参数，即 Adam 优化器设置的学习率为 0.0001，权重衰减率为 0.0005。在所有实验中，模型训练的迭代次数都为 20000 次，每一次训练迭代时

都会使用训练集一个批次的数据样本更新模型的参数，然后在可见验证集和不可见验证集的所有数据样本上测试当前模型的导航性能。

在 contrast-visionVAE-follower 模型的实验中，跨模态对比学习模块中的队列Q的长度K设置为 1000。

4.5.2 模型性能评估指标

R2R 任务与其他一些需要靠人的主观评价判断性能好坏的视觉或语言任务相比，它最大的优势之一是，任务成功或失败是可以使用数值进行定量衡量的。导航误差（navigation error）被定义为导航图G中 agent 执行任务的最终停止位置v_t与目标位置v^*之间的直线最短路径距离。如果导航误差数值小于 3m，则可以认为此次导航任务是成功的。设置 3m 为导航误差的阈值是考虑到在导航图G中，3m 大约是一个视点的误差范围，这个误差是可以接受的。要注意的是，评估 agent 在执行任务过程中移动的整个完整轨迹是不必要的，因为人类在进行实际导航时所给出的很多自然语言指令是没有指定某种特定的必须应该采取的路径的。评估指标的核心是要定量地衡量 agent 在认为自己已经到达目标位置后选择执行结束动作以结束该次任务的可靠性和准确性。

更详细的，有几个重要与常用的指标来评估 agent 执行任务的好坏。即成功率（SR），导航误差（NE），oracle 成功率（OSR）和 oracle 导航误差（ONE）。其中成功率和导航误差是最直观和最重要的评估指标，而 oracle 成功率和 oracle 导航误差则是一般用于在训练模型时监控训练过程的性能变化。

（1）导航误差：定义为 agent 的最后停止位置与目标位置之间的直线路径距离；

（2）成功率：在某一次具体的导航任务中，如果 agent 最终的停止位置与目标位置的直线最短路径距离小于 3m 的阈值，则认为该次任务是成功的。当执行多次导航任务时，所有导航任务中成功的次数与导航任务总数之间的比值即为成功率；

（3）oracle 导航误差：定义为 agent 在执行某次任务时移动的完整路径的所有节点与参考路径的最后一个节点，即目标位置之间的最小距离。与导航误差更关注任务的最终停止位置的误差不同，oracle 导航误差更关注的是导航任务的整体过程的误差，对导航任务的精度要求比导航误差低；

（4）oracle 成功率：当 agent 在执行某次任务时移动的完整路径的所有节点中存在多于一个节点与参考路径的最后一个节点，即目标位置之间的距离小于 3m，则认为该次导航任务为 oracle 成功。当执行多次导航任务时，所有导航任务中 oracle 成功的次数与导航任务总数之间的比值即为 oracle 成功率。

4.5.3 实验结果分析

在服务器上启动已经安装好的 Matterport3D 模拟器，进入到 follower 模型或 contrast-visionVAE-follower 模型对应的代码存放的相应文件目录中，在服务器命令行中输入以下指令，即可开始模型的训练过程：

python3 tasks/R2R/train.py

在进行多组实验后得到的 follower 模型和 contrast-visionVAE-follower 模型相应的实验结果如表 4-2 所示：

表 4-2 实验结果表

模型	实验随机数种子设置	可见验证集导航成功率（%）	不可见验证集导航成功率（%）	可见验证集导航误差（m）	不可见验证集导航误差（m）
follower	1	53.1373	28.6079	4.6982	7.0784
	42	54.2157	28.4376	4.7655	7.0862
	128	49.4118	26.9051	4.9361	7.4445
	random	49.9020	29.1188	5.0740	6.9098
contrast-visionVAE-follower	1	54.2157	32.3116	4.4970	6.6377
	42	57.6471	31.1622	4.5256	6.5647
	128	55.1961	31.5028	4.5072	6.5484
	random	55.9804	32.8651	4.5921	6.7728

在模型的代码中，各种随机数的初始化可以使用随机数种子进行人为的设定，这样可以保证实验结果的可复现性，随机数种子参数值相同的 follower 模型和 contrast-visionVAE-follower 模型的实验互为对照实验。当随机数种子参数没有人为给定确定数值时，由计算机随机赋值。

由实验结果可知，在相同的实验设置下，改进后的 contrast-visionVAE-follower 模型无论是在可见验证集上的导航成功率还是在不可见验证集上的导航成功率都比原来的 follower 模型要有所提高，同时，可见和不可见验证集上的导航误差也有所减小。不考虑实验误差的偶然性，contrast-visionVAE-follower 模型

在不可见验证集上的导航成功率的提升效果与可见验证集相比更加显著，同时，contrast-visionVAE-follower 模型在不可见验证集上的导航误差的减小效果与可见验证集相比更加显著。

对于 contrast-visionVAE-follower 模型在不可见验证集和可见验证集上的性能提升差异，从模型结构上给出合理的推测与分析。一方面，增加的跨模态对比学习模块在训练的过程中学习语言指令和视觉图像序列跨模态信息的匹配关系，拉近相似跨模态信息在隐空间中的距离，拉远不相似跨模态信息在隐空间中的距离，使得模型能够更加有效地编码语言和视觉信息。另一方面，增加的视觉变分自编码器模块在训练过程中能学习到视觉图像序列的概率分布情况，并重建出相似的新视觉图像序列，因此即使每一次训练的时候输入的是相同的数据样本，但是由于输入到动作预测解码器中的是经过概率分布采样得到的重建的相似但不完全相同的视觉特征表示，因此效果相当于增加了有限数据样本的多样性，使得模型在训练的过程中能够根据已知的视觉图像信息合理地推测与已经观察过的视觉信息不同但相似的未见过的视觉信息，提升模型的泛化性能。因此，contrast-visionVAE-follower 模型在不可见验证集上进行测试时，由于具有一定的对不可知视觉信息的推测能力，所以性能提升显著。

综合来看，改进后的 contrast-visionVAE-follower 模型的导航性能要优于原来的 follower 模型。

4.6 本章小结

本章实验所使用的数据集是房间到房间任务常用的 R2R 数据集。首先对数据集的划分进行了说明，然后介绍了实验所使用的 Matterport3D 模拟器和实验环境的配置方法，之后介绍了训练模型时相关参数的设置和模型的性能评估指标，最后通过对 follower 模型和 contrast-visionVAE-follower 模型各自的多组实验结果的分析说明本文改进后的 contrast-visionVAE-follower 模型的导航性能要优于原来的 follower 模型。

第五章　总结与展望

本文针对视觉语言导航（VLN）领域中具体的房间到房间（Room-to-Room，R2R）任务进行了相关研究。首先概述了本课题的研究背景和研究意义以及相关研究者在 R2R 任务方面的研究现状，然后介绍了研究工作中用到的相关理论和具体的 speaker-follower 模型中的 follower 模型，紧接着对本文改进的 contrast-visionVAE-follower 模型进行了详细的设计说明，最后通过多组对照实验验证了本文研究的改进模型具有较好的导航性能。

本文的主要研究工作概括为以下几点：

第一，对相关研究者提出的 speaker-follower 模型中的 follower 模型进行了较为详细的分析，梳理了视觉语言导航领域中的 R2R 任务的数据处理流程。

第二，基于 follower 模型存在的不足，改进出本文研究的 contrast-visionVAE-follower 模型，引入跨模态对比学习模块和视觉变分自编码器模块，以提升导航任务的性能。

第三，进行了多组对照实验，基于明确的评估指标进行定量分析，得出原始 follower 模型和改进后的 contrast-visionVAE-follower 模型的性能差异情况，并对改进模型性能提升的原因进行了较为合理的分析。

总体来看，本研究所使用的模型属于监督学习模型，需要在已经标注好的数据集上进行训练，然而真实、高质量的现实环境数据是非常稀缺的，在现实环境中丰富的视觉信息具有多样性，在有限数据集上训练良好的导航模型当迁移到未知的环境中进行测试时往往会出现性能显著下降的问题，泛化性能较差。如何使用大量的未标注的视觉数据进行导航模型的无监督预训练将是未来研究的一个重要方向，现有的纯视觉或纯语言的无监督与训练模型在计算机视觉或自然语言处理领域已经表现出了巨大的学习能力，然而对于视觉语言导航这样的多模态任务来说，需要结合视觉和语言等多种模态信息进行无监督预训练的新模型和新方法，以进一步提高模型对多模态信息的理解能力和导航性能。

参考文献

[1] Anderson P, Wu Q, Teney D, et al. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 3674-3683.

[2] Ma C Y, Lu J, Wu Z, et al. Self-monitoring navigation agent via auxiliary progress estimation[J]. arXiv preprint arXiv:1901.03035, 2019.

[3] Ma C Y, Wu Z, AlRegib G, et al. The regretful agent: Heuristic-aided navigation through progress estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 6732-6740.

[4] Ke L, Li X, Bisk Y, et al. Tactical rewind: Self-correction via backtracking in vision-and-language navigation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 6741-6749.

[5] Huang H, Jain V, Mehta H, et al. Transferable representation learning in vision-and-language navigation[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 7404-7413.

[6] Zhu F, Zhu Y, Chang X, et al. Vision-language navigation with self-supervised auxiliary reasoning tasks[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 10012-10022.

[7] Fried D, Hu R, Cirik V, et al. Speaker-follower models for vision-and-language navigation[J]. Advances in Neural Information Processing Systems, 2018, 31.

[8] Hong Y, Rodriguez-Opazo C, Wu Q, et al. Sub-instruction aware vision-and-language navigation[J]. arXiv preprint arXiv:2004.02707, 2020.

[9] Agarwal S, Parikh D, Batra D, et al. Visual landmark selection for generating grounded and interpretable navigation instructions[C]//CVPR workshop on Deep Learning for Semantic Visual Navigation. 2019, 2.

[10] Parvaneh A, Abbasnejad E, Teney D, et al. Counterfactual vision-and-language navigation: Unravelling the unseen[J]. Advances in Neural Information Processing Systems, 2020, 33: 5296-5307.

[11] Fu T J, Wang X E, Peterson M F, et al. Counterfactual vision-and-language navigation via adversarial path sampler[C]//European Conference on Computer Vision. Springer, Cham, 2020: 71-86.

[12] Yu F, Deng Z, Narasimhan K, et al. Take the scenic route: Improving generalization in vision-and-language navigation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020: 920-921.

[13] An D, Qi Y, Huang Y, et al. Neighbor-view enhanced model for vision and language navigation[C]//Proceedings of the 29th ACM International Conference on Multimedia. 2021: 5101-5109.

[14] Liu C, Zhu F, Chang X, et al. Vision-language navigation with random environmental mixup[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 1644-1654.

[15] Sun Q, Zhuang Y, Chen Z, et al. Depth-Guided AdaIN and Shift Attention Network for Vision-And-Language Navigation[C]//2021 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2021: 1-6.

[16] Li X, Li C, Xia Q, et al. Robust navigation with language pretraining and stochastic sampling[J]. arXiv preprint arXiv:1909.02244, 2019.

[17] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[18] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[19] Hao W, Li C, Li X, et al. Towards learning a generic agent for vision-and-language navigation via pre-training[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 13137-13146.

[20] Majumdar A, Shrivastava A, Lee S, et al. Improving vision-and-language navigation with image-text pairs from the web[C]//European Conference on Computer Vision. Springer, Cham, 2020: 259-274.

[21] Hong Y, Wu Q, Qi Y, et al. A Recurrent Vision-and-Language BERT for Navigation. arXiv 2021[J]. arXiv preprint arXiv:2011.13922.

[22] Qi Y, Pan Z, Hong Y, et al. Know what and know where: An object-and-room informed sequential bert for indoor vision-language navigation[J]. arXiv e-prints, 2021: arXiv: 2104.04167.

[23] Cheng J, Dong L, Lapata M. Long short-term memory-networks for machine reading[J]. arXiv preprint arXiv:1601.06733, 2016.

[24] Kingma D P, Welling M. Auto-encoding variational bayes[J]. arXiv preprint arXiv:1312.6114, 2013.

[25] He K, Fan H, Wu Y, et al. Momentum contrast for unsupervised visual representation learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 9729-9738.

[26] Chang A, Dai A, Funkhouser T, et al. Matterport3d: Learning from rgb-d data in indoor environments[J]. arXiv preprint arXiv:1709.06158, 2017.

致　谢

光阴似箭，日月如梭，转眼间大学本科四年的学习时光已经接近尾声，我在人生的这个重要阶段中，得到了许多老师、同学和朋友们的帮助与支持，我一直心存感激。

首先，感谢我的指导教师刘老师，在毕业设计期间，刘老师一直给予我亲切的关怀和悉心的指导。感谢学校里曾经教导过我的所有老师，你们严谨的治学态度一直激励着我并将使我受益终身。同时，感谢我的师兄和师姐们，在我遇到困难的时候热情地帮助我，在我迷茫的时候耐心地开导我。我还要感谢我在大学里所结交的各位朋友，谢谢你们在生活上对我的关心和照顾，与你们在一起的日子将会成为我人生中一段美好的回忆。最后，我要感谢我的家人，感谢他们对我一直以来的理解与支持，做我坚实的后盾，让我可以勇敢地在大学里去尝试和锻炼自己。

最后，引用《送东阳马生序》中的一段话，总结自己的四年大学生活：“同舍生皆被绮绣，戴朱缨宝饰之帽，腰白玉之环，左佩刀，右备容臭，烨然若神人；余则缊袍敝衣处其间，略无慕艳意。以中有足乐者，不知口体之奉不若人也。盖余之勤且艰若此。”