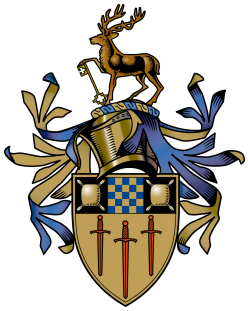
Multi-Task Autonomy For Robotics

Bian Xihan

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey



Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

September 2022

Abstract

Robotic technology is integrating into more aspects of our society. The ability to perform multiple different functions in various environments is an important research challenge. Recent developments in multi-task learning in Artificial Intelligence (AI) and machine learning could enable a new generation of robotic technology. This thesis aims to explore techniques which combine AI with robotics to enable generalization across a range of environments.

As the level of abstraction possible in deep learning has grown a very promising avenue of research has become feasible: hybrid systems. In these approaches, an AI system is used to undertake high-level planning and strategisation, while the execution of the plans is undertaken by classical non-learned approaches. This enables a mutual compensation of AI and traditional robotic systems, as the AI agents excel at generalisation for high-level problems and long-term goal-oriented planning, while the robotics techniques specialise in precise and repeatable operations [24]. This thesis attempts to build upon this and further explore the multi-task-enabled AI agent and its potential in robotic tasks.

First, general-purpose robots need to be able to work in multiple different environments. Even when performing similar tasks, different behaviour should be deployed to best fit the current environment. In this thesis, we propose a new approach to navigation, where it is treated as a multi-task learning problem. This enables the robot to learn to behave differently in visual navigation tasks for different environments while also learning shared expertise across environments. We evaluate our approach in both simulated environments as well as real-world data.

Following this, we introduce a new perspective for learning transferable content in multi-task imitation learning. Humans are able to transfer skills and knowledge to new tasks. If we can cycle to work and drive to the store, we can also cycle to the store and drive to work. We take inspiration from this and hypothesize the latent memory of a policy network can be disentangled into two partitions. These contain either the knowledge of the environmental context for the task or the generalizable skill needed to solve the task. This allows improved training efficiency and better generalization over previously unseen combinations of skills in the same environment, and the same task in unseen environments. We used the proposed approach to train a disentangled agent for two different multi-task IL environments. In both cases, we outperformed the SOTA by 30% in task success rate. We also demonstrated this for navigation on a real robot, including the visual navigation task of the previous chapter. This demonstration was also ported to a real robot.

This ability to generalize to previously unseen tasks with little to no supervision is a key challenge in modern machine learning research. Researchers often rely on reinforcement and imitation learning to provide online adaptation to new tasks, through trial and error learning. However, this can be challenging for complex tasks which require many timesteps or large numbers of subtasks to complete. These “long horizon” tasks suffer from sample inefficiency and can require extremely long training times before the agent can learn to perform the necessary long-term planning. Therefore, we finally introduce CASE which attempts to address these issues by utilising adaptive “near future” subgoals. These subgoals are re-calculated at each step using compositional arithmetic in a learned latent representation space similar to the previous chapter. In addition to improving learning efficiency for standard long-term tasks, this approach

also makes it possible to perform one-shot generalization to previously unseen tasks, given only a single reference trajectory for the task in a different environment.

When we combine the works in this thesis, these contributions provide insights for a better multi-task generalization framework for long and complex robotic planning. The contributions centre on multi-task learning and crossover to robotic navigation, reinforcement/imitation learning, and generalization. The experiments in this thesis demonstrate that an AI-controlled robot can perform simple tasks as well as a non-AI robot, while also handling more difficult tasks. This thesis presents a general direction for robotic AI capable of handling multiple complex tasks and sequential generalization. This enables better autonomy in both AI and robotics in the future.

Key words: Reinforcement Learning, Imitation Learning, Multi-task Learning, Generalization, Robotics

Email: xb00043@surrey.ac.uk

WWW: https://www.surrey.ac.uk/people/xihan-bian

Acknowledgements

I would first like to express my gratitude to my supervisors Dr. Simon Hadfield and Dr. Oscar Mendez, who not only guided me throughout out this project but also supported me as friends. I couldn't have done this without their help. I also would like to thank Dr. Sai Gu, I wouldn't have had this amazing opportunity if not for his firm recommendation and support. I would like to thank the members of CVSSP, who have been so friendly and helpful throughout my entire time in Surrey.

I would like to thank my parents Prof. Wenjuan Guo (郭文娟) and Prof. Wenyang Bian (卞文阳) and the rest of my family who are always so loving and encouraging.

I would like to thank my friends Ye Ling (凌烨) and Nong Xiaoqi (农晓琪) for accompanying me through this journey. I would also like to thank Zhang Yanpeng (张雁鹏), Zhang Lianpin (张连聘) and all of my friends. They supported me to chase after my dream.

Declaration

This thesis and the work to which it refers are the results of my own efforts. Any ideas, data, images or text resulting from the work of others (whether published or unpublished) are fully identified as such within the work and attributed to their originator in the text, bibliography or in footnotes. This thesis has not been submitted in whole or in part for any other academic degree or professional qualification. I agree that the University has the right to submit my work to the plagiarism detection service TurnitinUK for originality checks. Whether or not drafts have been so-assessed, the University reserves the right to require an electronic version of the final document (as submitted) for assessment as above.

The work presented in this thesis is also present in the following manuscripts:

- Bian Xihan, Oscar Mendez, and Simon Hadfield. Robot in a china shop: Using reinforcement learning for location-specific navigation behaviour. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5959–5965. IEEE
- Bian Xihan, Oscar Mendez, and Simon Hadfield. Skill-il: Disentangling skill and knowledge in multitask imitation learning
- Bian Xihan, Oscar Mendez, and Simon Hadfield. Generalizing to new tasks via one-shot compositional subgoals

Signed:

Bian Xihan 卞西晗

Date: 2022/10/01

Contents

Nomenclature

MERLIN Multi-environment Reinforcement-Learning in Navigation

SKILL-IL Skill and Knowledge Independent Latent Learning

CASE Compositional Adaptive Subgoal Estimation

ICRA The International Conference on Robotics and Automation

IROS The IEEE/RSJ International Conference on Intelligent Robots and Systems

AI Artificial Intelligence

SOTA State-Of-The-Art

RL Reinforcement Learning

IL Imitation Learning

SLAM Simultaneous Location and Mapping

CNN Convolutional Neural Network

CPV Compositional Plan Vectors

GAN Generative Adversarial Network

VAE Variational Auto Encoder

TRPO Trust Region Policy Optimization

PPO Policy Optimization Algorithm

AC Actor-Critic

A3C Asynchronous Advantage Actor-Critic

PNN Progressive Neural Network

KL Kullback-Leibler Divergence

FS Fixed Sampling

DL Dynamic Loss Weighting

SP Higher Skill Partition

KP Higher Knowledge Partition

PCA Principal Component Analysis

CI Current State Image

HRL hierarchical reinforcement learning

Symbols

Introduced in Chapter 3

I_c	Current Observation
I_t	Target Observation
ω_τ	Attention Weight Matrix
a	Action
A_i	Action Probabilities
O_C	Composed Current State Observation
$\mathcal{F}$	Feature Embedding Function
s	State in an Episode
n	Number of Sub-Networks
π	Learned Policy
E	Actor-Critic Branch of the RL Network
ω_V	Overall Weight for the Value Function
ω_τ	Attentive Distribution Weight
p_{stp}	Timestep Penalty
p_{cr}	Penalty for Hits Obstacles
l_{lm}	Episode Length Limit
R	Final Reward
G	Square Grid Environment
$\mathcal{U}$	Uniform Random Sampling Noise

Introduced in Chapter 4

O_t	Current State Observation
t	Timestep
$\vec{v}$	Compositional Task Embedding
$\vec{u}$	Compositional Task Embedding
g_ϕ	Encoding Function for Task Embedding
d_θ	Decoder Network of the VAE
L_δ	Reconstruction Loss
L_a	Policy Loss
L_H	Compositionality Loss
L_P	Similarity Loss
L_R	Regularization Loss
$L_\mathcal{G}$	Dynamical Loss
$\mathcal{S}$	Skill Training Set
$\mathcal{K}$	Knowledge Training Set
$\mathcal{N}$	Normal Training Set
$\mathbb{E}$	Goal Generative Process of the VAE
D_{KL}	Kullback-Leibler Divergence

Introduced in Chapter 5

C	Constraints Matrix
z	Zero Matrix
k	Subgoal Lookahead Parameter

List of Figures

List of Tables

Chapter 1

Introduction

The application scenarios for modern robotics are expanding as the technology evolves. Robots are given more tasks in more aspects of life, and are required to face more difficult challenges. They have progressed from vacuuming a room to delivering takeout food, ranging from working on assembly lines to managing an entire factory. Currently, application scenarios for robotic technology are often industry-related, but more service-related applications are developed every day. For robots to perform in a wider range of scenarios, they need support for multi-tasking in both hardware and software. In hardware, the research effort has been in the direction of general-purpose robotic AI solutions which emphasize the physical mobility and capability of robots, specifically legged robots and operational limbs. In software, the robot is expected to be able to distinguish between multiple tasks and perform them accordingly. This requires an understanding of the contextual meaning of the tasks and the knowledge for executing them. In this PhD, we aim to expand the capabilities of AI in robotics, especially in generalization and solving complex tasks with reinforcement learning and imitation learning, to enable general-purpose robotics with multi-task learning and generalization.

To accomplish such a task, the first requirement is enabling the robots to solve simple tasks such as navigation or single-action operations. Regarding general-purpose robotic AI solutions, neural network controlled bipedal and quadrupedal robots have been studied since the 90s. These robots can theoretically navigate and operate in a wide range of environments but require complex motion planning and a dynamically balanced gait. Their physical capabilities allow them to be adapted to a wide range of work environments, especially when compared to wheeled

robots or stationary robots. However, while there have been studies done on the topic of stair climbing and mobility in narrow spaces, gait control and navigation will require specialized or high power consumption equipment, which implies additional weight and less payload allowance, or shorter operation time. Moreover, the complicated nature of motion control also implies a high number of learnable parameters in continuous space, and the training of such a module will be extremely difficult. Therefore, it would be challenging to train an agent that excels in these basic tasks while also being capable of solving complex reasoning tasks.

We circumvent this predicament by utilizing a concept similar to hierarchical reinforcement learning (HRL). HRL is a framework that involves learning and decision-making at multiple levels of abstraction. At the highest level, there is a goal-setting module that determines what the agent should be trying to accomplish, such as navigating to a certain position, finding and picking up certain items, etc. At lower levels, there are specialized modules that are responsible for executing specific tasks necessary to achieve the overall goal, such as moving the robot forward by a certain distance and turning the joins on its manipulator. We utilize this idea to develop robotic systems, where the high-level decisions are made by an AI agent, and the lower-level execution is "out-sourced" to lower-level expert network controllers, or other specialist software solutions, which can handle the specific task with higher performance. Additionally, we can utilize traditional methodologies in fields such as motion control and Simultaneous Location and Mapping (SLAM) navigation as part of the lower-level control. These algorithmic methods can calculate rather than predict the actions of the robot to achieve the optimal deterministic solution with high confidence. The key benefits of these methods include precision, repetitive reliability, and robustness.

1.1 Challenges with robotic AI

To keep a high precision in robotic action is challenging, especially in locomotion and navigation. Be that grabbing an apple or moving to a specific spot on the map, researchers spend decades on making these operations as precise as possible and eliminating as many errors as possible. However, this precision is difficult for AI to replicate, as it often comes with an enormous amount of additional training and over-fitting. This is counter-productive as the same function can likely be performed by a lower-level control solution with better results and faster computation.

Moreover, achieving precision in a continuous space would require an escalation in training data and time, which does not necessarily translate to run time efficiency or speed. To achieve precision, a large network is often required. This means more learned parameters and layers to push forward during run time. In robotic applications, the onboard device often has a small memory size and requires a low computation cost. A large network dedicated to precision on a specific task is impractical in terms of reaction speed as well as power consumption.

Reliability plays a critical role in industrial settings. Industrial machinery often has fastidious requirements for reliability, meaning that a new device must be able to operate continuously for thousands of hours before any faults may occur. This requirement is a basic one for most industrial production machinery, and most lower-level expert techniques can meet this requirement. However, an AI-based solution often has several equally likely predictions, which leads to uncertainty in the input-output correspondence. In terms of reinforcement learning and imitation learning, the action prediction made by the network often has a close second choice in probability. This in turn compromises the repetitive reliability of the system.

Robust error handling is a critical capability for expert solutions. Traditionally, error handling and recovery are typically hand-crafted which ensures a quick solution whenever a known error occurs. However, these solutions are usually one-size-fits-all, such as shutdown, cut power, or restart. When the problem becomes more complex and requires generalizing from past experience, AI-based approaches started to demonstrate their advantages. One category of study in machine learning is fault prediction, where the AI network is trained to predict the possible error occurrence in an industrial system. These studies often rely on years of collected data, predicting known errors, and are often capable of generalizing to unknown ones.

Comparatively, the AI-based method excels in reasoning and contextual tasks such as identification, classification and evolving towards solving more complex tasks. From Deep Blue to Alpha Go, machine learning techniques which are cheaper, faster and better in complex tasks are invented at an astonishing speed. In the coming decade, we will get closer and closer to crossing the barrier of the invention of strong AI which has multiple purposes, and is capable of learning new knowledge and skill on its own. These tasks require a high degree of flexibility and generalization (in particular extrapolating from known situations to similar unknown situations) which is the area where AI excels. In the interest of improving the practicality of autonomous

robots being deployed in the field, we should make an effort towards unifying higher-level AI with lower-level experts.

1.1.1 Unifying AI and Automation

By unifying AI with lower-level expert software, we can harness the benefits of both approaches to create a composite architecture. This approach allows for higher-level decision-making AI to perform complex tasks and make plans, which can then be executed by lower-level expert software. This aligns naturally with reinforcement learning and imitation learning, where the AI agent outputs actions to achieve a long-term goal. Expanding this approach to an architectural scale allows us to control the entire robot using hierarchical reinforcement learning agents. To be more specific, we hope the robot programmed in this mixed architecture will have the AI deciding what the long-term goals or functions of the robot are, and how to execute the plan. Then, a library of task-specific AI or software will handle the robot to execute the steps outlined in the plan. This framework empowers us to focus on developing the multi-task learning and generalization capability of the AI agent, thereby taking a step towards general-purpose AI. With this approach, we can effectively solve increasingly complex tasks while still benefiting from the precision, reliability, and robustness of lower-level expert techniques. Once we demonstrate the practicality of this concept, we can further concentrate on complex multi-task learning and the generalization of the AI agent. We view this as an opportunity to further robotic research in collaboration with AI, representing a small step forward toward general-purpose AI.

1.1.2 Solving Complex Tasks

In the scope of this PhD, we focus on solving complex tasks using reinforcement learning and imitation learning. We will first develop upon the concept of AI controlling non-AI functions for robotic operation. Subsequently, we will delve into gaining a better understanding and learning complex tasks in multi-task learning scenarios. Additionally, we will dedicate effort to enhancing the generalization capability of these AI agents.

To define and learn a complex task, we must first establish a clear definition of what we mean by a task. In this work, a task is defined as a specific piece of work or labour that an agent must

complete within a finite space or environment, with a defined purpose and requiring a certain number of actions or interactions from the agent.

For instance, let's consider the task"*Bring me a cup of coffee*". This task defines the purpose of the agent as retrieving a cup of coffee. This task requires the robot to perform a series of labour: locate the coffee machine, navigate to the coffee machine, make coffee and bring it back. This labour has the implicit requirements of within a reasonable time and within the reasonable working space of the robot. (otherwise, the robot might bring back the coffee after 3 years, or navigate to the other side of the world to make coffee) A task may contain sub-tasks. A

Figure 1.1: *Make Coffee* is a sub-task of *Bring Back Coffee*, and *Make Coffee* is itself a complex task, as it contains multiple sub-tasks.

sub-task is a partition of a larger task and an be considered a task itself. it can also contain its own sub-tasks. The sub-tasks within a task may or may not have related, either in ordering or in reasoning. Sometimes, the sub-tasks are related implicitly through reasoning or timing. These sub-tasks are typically executed one at a time, only beginning another task after completing the current one.

In the coffee example, *locating the coffee machine*, *navigating to the coffee machine*, *making coffee* and *bringing it back* are all sub-tasks of the original task. Each of these subtasks also contains several sub-tasks, such as *opening the door to the kitchen*, *moving towards the coffee machine*, *checking if there's already coffee*, *getting water*, *getting a coffee pod*, etc.

While every task can almost infinitely separate into smaller sub-tasks, when a task is so simple

that it can be completed within a single expert instruction at the decision-making level, we consider this an action. Action is the basic building block of labour which completes a task. The form of action can sometimes be more complex than basic instruction, the acceptable level of complexity depends on the level of abstraction in decision-making and is often environment and task specific. For example, *picking up a coffee mug* is itself a complex task and requires multiple steps to complete. However, when considering the entire task of *bringing back coffee* at a high level of abstraction, *picking up a coffee mug* can be considered as an action rather than a subtask.

This is where we embed the lower-level expert technique into the architecture. By utilizing these techniques and treating them as actions, the AI model doesn't have to reproduce these works and can focus on more important decision-making. To navigate to the kitchen, the robot needs to perform SLAM navigation and path planning, and then actuate itself. These tasks are all done through expert methods, as they have mature solutions in their particular field. The AI agent would treat them as actions rather than tasks. This will make the training process much easier for the AI agent, as the action subsystem will utilize expert techniques to handle all the details. However, this complex action process does make the environmental dynamics far more complex for the RL agent, and model-based RL is no longer possible.

A complex task is a task which contains at least two sub-task, the task "*bring back a cup of coffee*" is a complex task. In a simplistic simulation of reality, we can treat its subtasks such as making coffee using a coffee machine as a simple action, the AI model only needs to know when to perform this action but does not know how to excite this action. Even with this simplification, the task is still a complex one. Fundamentally, to handle such difficult problems, robots should have an understanding of the different tasks and often a series of tasks in a sequence. AI which requires a large amount of data to train will have difficulty in dealing with a complex environment with multiple tasks as well as performing them in a logical order. This is especially difficult when the AI needs to operate in a dynamic environment, interacting and actively changing its surroundings. In the "*getting coffee*" example, the environment includes a kitchen, an office, and the hallway connecting them, but when the robot moves to another environment where there are stairs, or a cafeteria involved, the strategies required to complete the same task would change, and the robot's behaviour would also need to adapt to the corresponding environment. To overcome these difficulties of training an AI model for robots, it's possible to

Figure 1.2: The hardware setup greatly impacts the functionality and navigation technique deployed on a robot, while the skill of flying cannot be used for a wheeled robot, the map built by a drone can still be utilized.

rely on reinforcement learning and imitation learning. Reinforcement and imitation learning have a natural advantage in semi-supervised, situational, and continuous learning for long-term strategies.

The ability to operate in different environments with different behaviour entails the capability to multi-tasking. Multi-task learning has been a fascinating and fast-growing field in the reinforcement learning area. However, we still lack the understanding of what factors actually affect the learning of multiple tasks. We hypothesize the latent memory of a policy network is similar to a human's memory system which follows the PD paradigm. Procedural memory, or skill, is the memory required for the agent to perform a certain task in general. Declarative memory, or knowledge, involves memory specific to the environment the agent is operating in. In the running example, the capability of navigating to a certain location is a skill the robot has, and the path to the coffee machine is knowledge. When the robot is moved to another environment, like a cafeteria, the skill of navigation to a defined location is still preserved, but the knowledge of the map layout of the office and kitchen is no longer useful. If we switch the robot to a drone, the skill of navigation (on wheels) would be obsolete, but the knowledge of the layout of the building would still be useful. We hypothesize the latent memory of a policy network can be disentangled into two partitions each representing either skill or knowledge.

However, the inherent complexity of a task remains a challenge, as it often requires a large number of steps to complete an episode. This is particularly true for tasks with terminal-only

sparse rewards. The longer the average trajectory is, the broader we can expect an unbounded state space to become, which means lower sample efficiency.

The second challenge is generalization. In Imitation Learning, the data efficiency problem will often manifest as a relatively restricted set of expert trajectories. As such learning to perform a complex task often involves repetitive training on a small set of sample tasks. This can easily lead to over-fitting on the training task set or the specific training examples of the tasks.

To address these challenges in solving a complex task, one reasonable approach is to simplify the problem by breaking down the complex task into smaller, easier tasks. These simplified sub-tasks can be broken down the simplified sub-task into other sub-task until they can be easily learned. With the addition of utilizing the lower-level expert method as actions to complete sub-tasks. We expect this architecture to be capable of multi-tasking, utilising matured robotic software for simple tasks, and capable of solving complex tasks by utilising sub-tasking. We expect to relieve robots from the limitation of a singular environment, allowing robots to adapt to the diversity of modern social environments. In multitasking, we expect the agent can learn the difference between skill and knowledge, which would accelerate the learning process in multi-task learning, as well as increase the generalization capability of the agent. Finally, by breaking down a long complex task into sub-tasks with easy-to-learn, quick-to-achieve sub-goals, the AI agent will gain the ability to learn and solve complex tasks in a relatively short amount of time with a good generalization performance.

1.2 Motivation

This PhD aims to advance robotic capabilities by enabling the performance of multiple complex tasks at an expert level. The proposed approach involves utilizing reinforcement learning and imitation learning to develop an agent that can deconstruct large, complex tasks into smaller, easier sub-tasks. By successfully executing these individual sub-tasks, the agent can ultimately achieve a solution for the overall complex task.

To achieve the objective of deconstructing complex tasks, it is essential for the AI model controlling the robots to possess contextual understanding. This entails the ability to recognize and adapt to different environments, resulting in different behaviour strategies. Human behaviour

serves as an example, where we adjust our behaviour when entering different environments. For instance, we adopt a slow and careful approach to avoid collisions in a china shop, whereas walking down an empty hallway requires more speed than precision. Robots need to possess a similar ability to understand the context of their environment and adjust their behaviours accordingly. This enables them to perform better on multiple isolated tasks. As complex tasks can often be broken down into multiple sub-tasks executed in sequence, the subsequent stage of research will focus on the transition between tasks. Since each isolated task is associated with a corresponding expert network, the transition between tasks essentially involves a shift from one dominant expert network to another.

To enhance the transfer of learned knowledge between different tasks and improve generalization to unseen tasks, we naturally draw inspiration from human behaviour. Humans possess the ability to transfer skills and knowledge. When faced with a new problem, we can leverage our experience of solving similar problems in different environmental settings. For instance, we can drive a car in a new city without needing to re-learn how to drive. We simply need to familiarize ourselves with the layout of the new environment. This suggests that if we can drive to work, we should also be able to walk or cycle to work. Motivated by this observation, we hypothesize that the latent memory of a policy network can be disentangled into distinct partitions. Each partition would contain either the knowledge of the environmental context for a given task or the generalizable skill necessary to solve that task. By disentangling these components, we can potentially enhance training efficiency and provide the agent with better generalization capabilities when facing unseen combinations of skills and knowledge.

Once the robots have determined the optimal order to complete a set of given tasks, the final step of this PhD will be to enable the robots to break down a single complex task into multiple manageable sub-tasks. This ability which would allow the agent to generalize to previously unseen tasks with little to no supervision is a key challenge in modern machine learning research. It is also a cornerstone of a future “General AI”. Any artificially intelligent agent deployed in a real-world application must adapt on the fly to unknown environments. This can be challenging for complex tasks which require many timesteps or large numbers of subtasks to complete. These “long horizon” tasks suffer from sample inefficiency and can require extremely long training times before the agent can learn to perform the necessary long-term planning. The ultimate objective of this PhD research is to enhance the learning efficiency for standard long-term tasks

and achieve generalization to previously unseen tasks. By addressing these challenges, we aim to improve the overall capabilities of AI agents in real-world applications.

With this AI framework, the application scenarios of robots could be expanded, allowing faster development of AI-based robotics and making a step towards true multi-purpose robots. The summary of the objectives which set a path to the overall aims of this thesis are as follows:

1. Develop a multi-task agent, with emphasis on navigation. Through multi-task reinforcement learning, allows robots to perform visual navigation tasks in multiple environments without re-training.

2. Develop a cross-task agent. Allowing the robot to handle the transition between tasks' corresponding expert networks.

3. Develop a disentangling learning framework, which would improve training efficiency and generalization capability over unseen combination sets of skill and knowledge.

4. Develop a complex task-solving agent, which utilizes the concept of deconstructing the complex task into sub-tasks. This agent should have the ability to perform reasonably well in unseen tasks as the focus of this aim is generalization capability.

1.3 Contributions

The first technical chapter introduces the work of Multi-environment Reinforcement-Learning in Navigation (MERLIN), which propose a multi-task learning-based navigation technique. This agent enables the robot to learn to behave differently in visual navigation tasks for different environments while also learning shared expertise across environments. The work is published in The International Conference on Robotics and Automation (ICRA) 2021 with an oral presentation.

The second technical chapter consists of the work of Skill and Knowledge Independent Latent Learning (SKILL-IL), which introduced a new perspective for learning transferable content in multi-task imitation learning. This work attempted to disentangle the latent memory of a policy network into two partitions each containing the skill and knowledge learned during

training. This allows for better generalization capability as well as improved task performance. This work is published at The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022 with an oral presentation.

The third technical chapter introduces Compositional Adaptive Subgoal Estimation (CASE). This work utilizes the sub-goal approach in imitation learning to improve the learning efficiency for long-term complex tasks, as well as improve the generalization over unseen tasks. This work is presented at ICRA 2022 workshop.

1.4 Summary

In summary, the main contribution of this thesis is to advance the research on reinforcement learning and imitation learning within the context of multi-task learning in robotics. The subsequent chapters will follow a structured format, starting with a literature review where the relevant background theory and research will be discussed. Each section of the literature review will correspond to a specific technical chapter. The three chapters followed will be the technical chapter, explaining in detail the research and its evaluation. The MERLIN contributions of Chapter 3 satisfy the first and second aims of the thesis. The SKILL-IL in Chapter 4 satisfies the second aim. Finally, Chapter 5 addresses the final aim of CASE. The final chapter concludes the research work in this thesis, it presents a summary of the contributions and the limitation of the research, and clarified possible future work directions. The final chapter will serve as the conclusion of the research work presented in this thesis. It will provide a summary of the contributions made, discuss the limitations of the research, and outline potential directions for future work.

Chapter 2

State-of-the-Art

In order to explore multi-task autonomy for robotics, this PhD relies on a broad range of topics, both fundamental and task specific. This starts with the problem of visual navigation as an entry point to creating a system where the AI agent is controlled but the traditional robotic methods are in place to handle the actions. The current research focus involving visual navigation often emphasises on semantic-based mapping and navigation, as a semantic map is more meaningful with condensed information compared to landmark or featured-based visual navigation.

To seek a better understanding of multi-task learning in an Reinforcement Learning (RL) and imitation learning framework, we make effort to disentangle the learned latent space with inspiration from psychological studies. More recent studies on RL focus on the explainability of the agent and solving complex tasks with unpredictable variations in the environments. Researchers are also making effort in inverse reinforcement learning which seeks the optimal reward function rather than the optimal policy. Imitation learning and reinforcement learning are both still making progress in data efficiency and generalization, which remain the biggest challenges. Multi-task learning researches focus on conquering the challenge of scalability and making attempts with combinatorial generalization.

Finally, we further export the complex multi-task solving with sub-goal and compositional vectors. The sub-goal technique is often applied to navigation problems, but some researchers have applied the idea to reinforcement learning and imitation learning as the sub-goal provides an easily learned target which eases the learning process. Compositional vectors are often used with complex tasks, especially in robotic-related tasks and combinatorial generalization.

This chapter introduces some state-of-the-art studies in these fields and provides context for our work. We will first explore the literature on deep reinforcement learning and imitation learning, then investigate the literature around sub-goal estimation for complex multi-task learning, and lastly the literature on the specific topic of visual navigation.

2.1 Deep Reinforcement Learning

Reinforcement learning refers to the method of machine learning in which an AI agent performs actions to maximize reward gain over a long-term episode. An agent is expected to learn strategies to obtain better results and form a policy which guides the agent's actions. Similarly, imitation learning also learns an action policy by mimicking an expert demonstrator. This differs from supervised learning as reinforcement learning focuses on making continuous sequential decisions that depend on the current input and the previous states, versus supervised learning which makes independent predictions for each input. This long-term planning capability fits well with robotic applications, as we expect the agent to perform complex tasks which require a long-horizon strategy. This section will review the current works in the field of reinforcement learning and imitation learning.

2.1.1 Reinforcement Learning

The basic concept of RL is to maximize the long-term cumulative reward through trial and error. The AI model, or agent, is placed in an environment which it can interact with and obtain a reward for its actions. The learning process of RL is unsupervised and is thus suitable for any problem without pre-labelled data or which requires long-term strategies. The most common RL methods are value-based approaches, such as Q-learning [110]. In Q-learning, the agent establishes and maintains a lookup table of expected reward value for specific actions it takes in any given state. The agent will be able to take the action that has the largest Q-value for the maximum expected reward. As the Q-value table requires discrete states, Q-learning and other value-based methods are often unable to handle continuous actions and states. To resolve this, another approach is policy-based methods such as policy gradient where a policy estimates the action from the current state [37]. As the policy can continuously compute the agents' actions,

this would enable the agent to handle continuous actions and time.

A typical policy-based RL algorithm is Shulman's 2015 work on Trust Region Policy Optimization (TRPO)[92]. This is a classic RL algorithm that utilizes a different approach. Trust region methods use a model function to estimate the objective function, and by optimizing the model function, perform better actions. The size of the policy update is constrained to monotonically decrease to ensure convergence. Schulman's later work in the Proximal Policy Optimization Algorithms (Policy Optimization Algorithm (PPO) and PPO2) improved upon Trust Region Policy Optimization (TRPO) [93]. These algorithms use a "surrogate" objective function to determine the next action while interacting with the environment. It assumes that with similar state input, the agent should take similar action, which lowers the rate and necessity of re-sampling. Instead of constricting the model functions, PPOPPO applies a penalty to policies that differ from the objective function.

Another RL algorithm with a combined approach is the work of Mnih et al. in the Actor-Critic (AC) series of techniques [74]. These algorithms combine the Q-learning and policy gradient by creating a policy-based actor. This actor chooses actions and then a value-based critic will score these actions. This allows the handling of both discrete and continuous problems, as well as updating more regularly for better learning efficiency. In the more advanced version: Asynchronous Advantage Actor-Critic (A3C), the algorithm allows asynchronously update to and get updated by the main policy by multiple agents, this advancement greatly increased computation and sampling efficiency, allowing for faster training given the growing computing power. The multi-agent support also enabled RL in multi-task learning, allowing one agent to learn multiple tasks simultaneously. Due to its asynchronous nature, it is an ideal fit for multi-task learning by allowing agents to learn multiple tasks simultaneously, and update to a single main policy. However, these techniques suffer from high sample complexity and brittleness to hyperparameters. Later work in [39] attempted to solve these issues by maximizing expected reward as well as action entropy. This ensures success in the task while taking providing as much variance in the selected actions as possible. This improves stability with respect to hyperparameters and accelerates training.

In a more recent development, reinforcement learning is treated as a form of representation learning. The learned latent is manipulated to better represent the information. The work of

De et al. [31] regularized the state representation during reinforcement learning. The work of Bellemare et al. [11] is based on geometric properties of the value functions and utilized the value function as an auxiliary task during reinforcement learning. It adapts the representation to minimize the approximation error of the value function. The work of Chandak et al. [20] decomposed a policy into one component which chooses actions in an abstraction representation space and one component which realizes these representations into actual actions The work of Carats et al. [118] addresses the data efficiency challenges of learning useful representations. They undertake representation learning with exploration through prototypical representations, which simultaneously serve as a summarization of the exploratory experience and observations.

2.1.2 Imitation Learning

Imitation learning is very much similar to reinforcement learning. The learning goal of imitation learning is to imitate the actions of an expert demonstrator and learn an action-based policy which performs the task as the expert would. We also explore the use of imitation learning in this PhD, along with reinforcement learning. Typical Imitation Learning (IL) approaches [90, 96, 27, 41] utilize a dataset of expert demonstrations to guide the learning process. Initially, imitation learning relied heavily on supervision. However, these approaches perform poorly with increasing episode length and have issues with generalization. To improve upon this, the work of Levine et al[61] uses expert trajectories to optimize the action policy rather than imitate the expert exactly. Later works move away from a passive collection of demonstrations. Instead, they exploit the active collection of demonstrations from experts. To this end [102] create an interactive expert which provides a demonstration as a response to the actions taken by the agent.

In more recent studies, imitation learning also has been found to have a strong connection with inverse reinforcement learning. Inverse reinforcement learning assumes an underlying optimized reward function and attempts to recover it, this mirrors the learning goal of imitation learning. The work of Jonathan et al. [44] extracts a policy with inverse reinforcement learning, making an analogy between imitation learning and generative adversarial networks which derive a model-free imitation learning framework. The work of Jiaming et al. [98] focused on multi-agent imitation learning. Their framework builds upon a generalized notion of inverse

reinforcement learning. This helps partially mitigate the issues with training data efficiency.

In more relevance to RL, [101, 100] aim to combine IL with RL, by using IL as a pre-training step for RL. The work of Cheng et al. [26] performs randomized switching from IL to RL during policy training to enable faster learning. The work of Murali et al. [78] uses expert trajectories to learn the reward function rather than the action policy. In this PhD, we use imitation learning to speed up the training of an agent which is evaluated in RL-based interactive environments, this greatly increases the data efficiency during training.

Additionally, the combination of imitation learning and multi-task learning has gained some attention in more recent studies, especially with relevance to robotics. The long-term policy planning capability of reinforcement learning and imitation learning combined with multi-task learning suits well with robotic applications' requirements. The work of Avi et al. [97] utilized the concept of a multi-task setting, treating a failed attempt at one task as a successful attempt at another, and leveraging the robot's own trials as demonstrations. Junhong et al. [116] proposed a multi-headed imitation learning framework where each task is treated as a sub-policy and shares information among related tasks by summing the activations of all sub-policies. The work of Zhao et al. [69] explores the one-shot imitation learning capability of ambitious multi-task setup in vision-based manipulation tasks. They integrate a self-attention model with a temporal contrastive model for better task disambiguation and more robust representation learning. Kipf et al. [54] utilized the idea of sequence segmentation by encoding sequential data which can be re-composed into new combinations to achieve generalization over longer tasks in unseen environments. These works all rely on an imitation learning framework, either utilizing multi-task learning concepts or applying the imitation learning method to the multi-task learning area. It shows the potential of this combination which we explore further during this PhD.

2.2 Disentanglement and Generalization

The biggest open challenges in multi-task RL are sample efficiency and generalization. In this thesis, we will explore task disentanglement as a solution to this. As such, we will now present a review of the current state-of-the-art in learned disentanglement.

2.2.1 Multi-Task Learning

Multi-task learning is generally defined as a model capable of performing more than one task after training. It is a rapidly growing field that appears frequently in both supervised learning as well as RL. Multi-task learning aims to improve the learning and performance of a model for several tasks. These tasks are not identical but can be related to varying degrees.

In multi-task learning, the relevance of tasks is an elementary factor in determining the best approaches: Feature-based models, first presented by Obozinski et al. [80], assume all related tasks can be represented with similar feature representations and can be acquired through the transformation of the original features. The model would learn this transformation for each task and compute the output. Later works often improved upon this approach with better optimization methods. Such as the work of Liu et al. [65] which uses l2, 1-norm for minimization and Lee et al. [60] which takes advantage of the structure of the data and prior information for regularized regression. However, they all require the task and data to be closely related, or in a similar structure.

In contrast, parameter-based models would encode the relatedness of tasks into model parameters. These approaches are better at handling tasks which are less related but still assume that the tasks are related enough to be represented through a parameter matrix[121]. Another common approach is a model consisting of two parts with one part a shared parametrization (the backbone) used by all tasks, and one part (the head) being unique to each task [99, 67]. The model is optimized by minimizing the training loss of these combined features across all the tasks. Instance-based approaches maintain a collection of data instances from all tasks and estimate the relevance of each stored instance to the new task at hand. The combined weights are then used to learn each task[1]. For a single robot, all the tasks it needs to learn will likely be similar or somewhat related [122]. It would be ideal to implement and train a single multi-tasking network that will be able to handle these tasks.

In multi-task reinforcement learning, numerous works have shown the ability of a single agent to perform at an expert level in multiple Atari games using deep Q-learning [75, 7]. These works, such as the work of Liang et al. [63], have proven reinforcement learning's potential in multitasking. In the RL environment, the model makes no assumption of the relatedness of tasks, which enabled many different approaches such as policy representation for each task and

regionalized policy clustering, the work of Kulkarni et al. [56] employs a hierarchical Bayesian approach to model the distribution over Gaussian process temporal-difference value functions for each task, and A3C-based deep reinforcement learning approach.

The challenges these approaches all face are negative learning and scalability. Negative learning refers to when agents 'forget' previously learned knowledge while learning a new task. The most popular solution to negative learning is through the use of a gating mechanism, where the network is only allowed to update part of itself during the training for each task. This idea is first proposed by Rusu et al. in the work of Progressive Neural Network (PNN) [87], where the network freezes itself and adds new resources for the new task. In the training process of the new task, the network stops updating its current weights and then widens all the nodes to provide new resources for learning new knowledge. Weights from previous tasks will be used to help the learning but won't be updated during the training session for the new task. This approach does provide a good solution to the problem of negative learning. However, it fails in scalability, due to increasing learnable parameters and poor sampling efficiency with respect to the increasing complexity of tasks.

To improve sampling efficiency, the work of Andrychowicz et al. [4] and Landolfi et al. [58] propose the use of a memory bank or model-based approach during sampling. For parameter size, a common solution to this is using model compression techniques for data efficiency, such as the work of Teh et al. in Distral [103], where the network shares a distilled policy that captures the common behaviours across all tasks and allows workers to solve their own task while staying close to the main policy. The work of Xihan et al. [115] disentangles the learned latent into subdomains containing different information regarding the skill for solving each individual task and the knowledge of the environment. Another approach by Oh et al. [81] presents a hierarchical architecture. Here a meta controller learns to use the acquired skills to execute a sequence of instructions after learning useful skills that solve subtasks.

In more recent studies, reasoning-related multi-task learning has gained some attention. A branch of the research focuses on solving complex tasks which can be decomposed into a sequence of several smaller sub-tasks. The work of Bozkurt et al. [16] introduces a model-free RL approach which maximizes the probability of satisfying the given linear temporal logic objective. The work of Xihan et al. [113] and Chane et al. [21] utilize the concept of sub-goals

to facilitate the learning of long-horizon complex tasks. These works solve the problem by treating the reasoning task as a series of sub-tasks and predicting intermediate states.

In multi-task learning, most prior research focuses on allowing the agent to perform multiple tasks without considering the varying similarity between tasks. The transfer of information between tasks is also not normally addressed explicitly. In [88], the agent was trained to learn how to perform a series of tasks in a sequence, sharing some weights in earlier layers and with independently trained heads added for each task. Yee et al. [104] expanded on this by maintaining a shared policy between the independently trained agents and distilling shared information from task-specific experts. Later research expanded this idea to hierarchical architectures. This allowed the agent to perform multiple sub-tasks as a part of a longer, more complex task. In the work of Shu et al. [95], the multi-level hierarchical policy decomposes a complex task into multiple levels of sub-task, with each having human instruction as a reference.

In hierarchical multi-task research, sub-tasks are often learned through linguistically categorised representations of a specific set of tasks. The representations used by these systems sometimes unintentionally explore the skill/knowledge paradigm. For example in the work of Andreas et al. [3], the policy sketches annotate tasks with sequences of subtasks. Each sub-task is explained as a combination of action (get, use) and a goal (wood, workbench). The work of Oh et al. [81], introduced the innovative "analogy" representation of subtasks. Here the target objects and the actions which can be applied to a target object are independent. This makes it possible to share information between tasks by forcing the representation of actions and objects to be independent in the embedding space. This approach allowed the agent to generalize to unseen combinations of actions and objects. The work of Xihan et al. [114] addressed a single type of task but focussed on learning different behaviours in different types of environments. The work of Devin et al. [33] introduced Compositional Plan Vectors (CPV), in which instead of learning the representation of each subtask, the network learns the embedding for a composition of a sequence of sub-tasks. This allows the decomposition of tasks without hierarchical or relational supervision. Our work bridges these different ideas, learning a latent space where not only can subtasks be composited, but where the skill and knowledge components of subtasks can be shared and permuted. This makes it possible to solve never before seen task combinations, as a step towards zero-shot IL and general AI. These sub-task-based researches, often unintentionally, comply with the procedural/declarative paradigm when choosing the hierarchical model of the

agent.

In this PhD, we initially developed a multi-branch gated network somewhat similar to PNN[87] but with soft blending and lifelong training of all branches training simultaneously on all tasks, instead of sequential training and freezing weights. This framework makes no assumption about the relatedness of tasks and mitigated negative learning effects through attention-based soft gating.

2.2.2 Skill and Knowledge Learning

Task disentanglement has its roots in psychological studies with amnesiac patients and non-amnesiac patients[111]. This has shown that human memory can be separated into procedural (skill-based) memory and declarative (knowledge-based) memory. Amnesiac patients are able to learn a set of spatial sequence-based tasks without awareness of the repetition. Neurotypical subjects show similar results and are able to solve the tasks without explicit declarative knowledge of the task. These studies have shown the different types of memory are not fully entangled in human learning.

Prior work in knowledge acquisition is often in the field of pedagogy, language acquisition, and neurobiology [105, 86]. Studies have determined the declarative memory system heavily depends on the hippocampus, medial temporal lobe and neocortical regions of the brain. Meanwhile, the procedural memory system involves a network based on frontal ganglia circuits. When consolidating new information, each system activates different regions of the brain.

In multi-task machine learning research, certain works would unintentionally avoid the entanglement of different types of memory. However, typical research would use a randomly generated environment and multiple types of tasks. This research often treats each task as a single entity [88] or as a unit part of an architecture[95]. In this chapter of this thesis, we attempt to explicitly disentangle the different types of latent subdomains within each task embedding. This enables us to improve data efficiency and share information across tasks in multi-task learning, as well as improve the understanding and explainability of multi-task learning.

2.2.3 Disentangled Representations

The state of the art for learning disentangled representations is dominated by VAE approaches. VAEs, or variational auto-encoders, are generative models which re-parametrize a latent space as a distribution to be sampled from. Compared to other popular generative models such as GAN, the VAE has been shown to be more robust and stable [18]. Each dimension of the latent representation learned by the VAE is generally considered an independent generative factor [30]. These elements in the representation can capture and isolate certain underlying factors without affecting other elements in the latent space. Although they are generally unsupervised, many authors have observed that the latent factors learned by a VAE often represent disentangled human interpretable concepts. A great deal of research has been done to explore the disentanglement of these learned representations further[18]. The work of Zheng et al. [123] disentangles the latent space into the label relevant and irrelevant dimensions for a single input. The work of Abdul et al. [6] augmented a standard VAE with an inverse-Wishart prior to encouraging independence in the learned latent dimensions. In the works of Kumar et al. [57] and Chen et al. [25], new objective functions are proposed to encourage disentanglement. In beta-VAE and the later work of Burgess et al. [42, 111], a variation of the VAE framework is proposed which balances the latent channel capacity and constraints with reconstruction accuracy. Allowing robust learning of disentangled representations without the trade-off in reconstruction quality.

However, these works generally rely on unsupervised learning making it challenging to impose prior knowledge about the generative factors or to share information between examples[43]. The work of Vowels et al. [107, 108] overturned this paradigm through a weakly-supervised approach which isolates domain knowledge in the training process of a gated VAE. This framework makes it possible to learn latent subdomains, by appropriately partitioning the training based on shared properties. The learning of the latent factors is still unsupervised, but additional losses are provided as a soft constraint to group the factors into subdomains. This method was shown to be more informative and has a better quality of disentanglement. Chapter 4 takes inspiration from this and proposes an algorithm to learn latent skill and knowledge subdomains explicitly.

2.2.4 Subgoal Search

For long-term composite tasks, another obvious form of disentanglement is the separation into subtasks. Although often not considered a subset of disentanglement, the subgoal search literature is rich. The most common use of Subgoal Search is in the context of path planning [12]. Here, specific landmarks or semantic regions are recognized as a subgoal for the robot to navigate towards. These tasks often use existing content within the observation as the subgoal, especially in vision-based navigation tasks. In [34], a two-level hierarchical approach is utilized to integrate model-free deep learning and model-based path planning with a low-level motion controller and a high-level path planner providing subgoals. The work of Savinov et al. [89] proposed a semi-parametric topological memory. This memory stores connectivity of locations corresponding to the nodes (treated as subgoals during path planning), and planning for navigation for goal-oriented tasks. The work of Liu et al. [66] introduced a visual planning method built on [89]. An energy-based graph connectivity function with a conditional VAE model generates images given a context describing the domain. Combined with hierarchical reinforcement learning, the work of Kim et al. [52] presents an RL framework with a reduced action space guided by landmarks and generates subgoals towards landmarks with more informative potential.

More recently there has been significant research on extending Subgoal Search beyond navigation tasks, particularly exploring the interaction with reinforcement learning. [9] is one of the earliest works combining hierarchical reinforcement learning with subgoal discovery. Here, high-level policies discover subgoals while low-level policies specialize in executing each subgoal. In a similar approach, [52] trains a high-level policy with a reduced action space guided by landmarks. [120] generalized this by looking for a sub-goal in a k-step adjacent region of the current state, within a general reinforcement learning environment. [62] builds upon this stable representation of subgoals, and proposes an active hierarchical exploration strategy which seeks out new promising subgoals with novelty and potential. All of these works discovered subgoals automatically, removing the need for hand-crafted subgoals. However, they all maintained the hard boundary between the subgoals. The work of Yu et al. [119] focuses on multi-subgoal robotic navigation tasks. Specifically, the framework builds a sub-graph network, which is a graph neural network with the history of all agents. Then subgoal selection is conditioned on this graph network where the next best node is selected.

The majority of the research in subgoal estimation focuses on optimizing the generation or assessment of a candidate sub-goal state. However, the work of Czechowski et al. [29] shows a simple approach for efficiently generating subgoals which are precisely k-steps-ahead in reasoning tasks. This is similar to the approach we propose in chapter 5 which chooses a possible "near future" subgoal, which adapts over time. This allows the policy network to focus on learning a more generalized skill for solving the overarching task rather than attaining any specific subgoal. This in turn improves the performance of our technique when encountering unseen tasks, and provides robustness by allowing recovery from errors.

Combining subgoal search with imitation learning is also getting attention in recent years, often combined with a hierarchical approach. Here, the idea of subgoals is utilized to generate an easy learning goal for the agent. The work of Paul et al. [83] uses a form of clustering on the expert trajectories to decompose the complex task into sub-goals. By learning to generate sub-goal states, the network obtains reward functions which direct the RL agent to move from one subgoal to another. However, the lack of compositionality in the model and the rigidity of the sub-goal prediction limit its generalization capability. The work of Wang et al. [109] proposed a subgoal-conditioned imitation learning framework without prior knowledge to encourage diverse representations among different subgoals. Applying this combination to a dialogue generation system with a multi-task learning approach, the work of Hsu et al. [46] attempts to build a multi-domain dialogue system with multi-task generative adversarial imitation learning. Their technique decomposes each of the complex tasks into several subtasks and hierarchically builds the policy.

2.3 Visual Navigation

Navigation is one of the most essential and fundamental operations in robotics and remains a major unsolved research question. It consists of a wide variety of studies extending from hardware design to deep learning based algorithms. Visual navigation is the field of study in navigation where the system mainly relies on visual input rather than other sensors. The natural advantage of visual navigation is visual information can easily be fed into a CNN-based deep learning framework. In practice, visual sensors are cheap and consume little power compared to active sensors like lidar. Additionally, visual navigation is very similar to how humans normally

behave, this opens many possibilities to mimic human behaviours. This section will focus on robotic navigation, more specifically the topic of visual navigation.

Traditional offline maps and obstacle-based navigation methods require a detailed global map of the entire environment. Most commonly a 2D occupancy grid, or virtual force field [51][14]. These global map approaches are generally extremely reliable, but collecting the map beforehand may be challenging, and they can fail if the map is not kept up to date. Because of this, SLAM algorithms were developed. This class of algorithm combines navigation with exploration to build a map online. In practice, SLAM may require sensor fusion for reliability and mapping can be computationally expensive. The cumulative error and environment complexity combined with uncertainties in sensors are huge problems for navigation methods in these approaches[82, 13]. Works in SLAM often don't address the exploration methods. A common practice in industrial applications is to combine the two approaches above, having a human who controls the robot during exploration and builds the map with SLAM techniques, and then using the map for autonomous navigation[59]. This approach eliminates a lot of complexity in exploration as well as adds stability to the map-building process. More recent mapless navigation techniques such as optical-flow-based and appearance-based matching are more closely related to the study of visual navigation.

Visual Navigation relies on vision as the primary source of information for navigation, compared to other navigation methods which rely on distance sensors. In map-based approaches for visual navigation, visual input is used mainly for landmark tracking. The algorithms detect landmarks on the camera input and track them in the following frames to determine the position of the robot[48, 40]. Another approach is to let the robot explore the environment and instead of a map, the robot builds a feature representation model of the environment[77]. Based on this idea, [24] explores navigation with topological maps and natural language instruction. The approach utilizes attention mechanisms to predict a navigation plan which is then executed with low-level actions. However, while classic map-based approaches are the foundation of artificial intelligence based visual navigation solutions, they still have difficulty handling large or complex spaces. These complexity and scaling problems are the root cause of current robotics being limited to a single working environment.

Mapless and AI-based navigation is becoming more popular. Mapless approaches [79], include

two main solutions: flow-based tracking (also known as visual odometry) and Appearance-based matching. Flow-based solutions estimate the motion of objects by tracking the motion of features throughout a sequence of visual inputs [45, 68]. Appearance-based matching solutions rely on prior knowledge from stored images of the environment [53]. The robot will try to match the current view with the stored images to localise and navigate [124, 8]. This idea of using learned features to predict the agent's position has been fundamental for AI-based localization and navigation. Notably, the work of Kendall in PoseNet [50] uses a convolutional network for real-time camera relocalization where the model is trained on labelled images of the environments. Recently, this idea has been extended to fully differentiable multi-hypothesis localization[72].

More reinforcement learning-based navigation techniques have emerged in recent years. The use of a semantic map with an attention mechanism is popular among these works. Graph convolutional networks are used for incorporating a prior knowledge graph into deep reinforcement learning for navigation tasks [117]. In target-driven visual navigation, the semantic map technique is employed by learning priors over the relative arrangement of objects in a scene [22]. The work of Ishihara et al. focuses on an end-to-end autonomous driving agent with multi-task learning capability[47]. Their approach also utilizes an attention mechanism for semantic segmentation and depth estimation. The concept of lifelong learning also applies to robotic navigation [64]. The lifelong learning scheme allows the robot to navigate in new environments while not forgetting previous ones with a self-improvement strategy which dynamically increases navigation performance. The work of Seifi et al. [94] improves on active visual exploration with self-attention and a contrastive stream on multiple tasks. The work of Morad et al. [76] utilized curriculum learning with deep reinforcement learning to navigate through unknown cluttered indoor environments to semantically-specified targets.

We approached the problem of multi-environment navigation through target-driven navigation, which is a branch of appearance-based matching. This approach employs the use of deep RL which does not require supervised training for landmarks or features. The work of Zhu et al. [124] presents a State-Of-The-Art (SOTA) target-driven visual navigation solution. The model they proposed is an actor-critic model which has a policy function for both the goal and the current state as input. This approach allows for better generalization and enables the model to train in a simulated environment through reinforcement learning. The work of Mayo et al. [70]

also focuses on target-driven visual navigation with a spatial attention probability model. This encodes semantic information about observed objects as well as spatial information about their placement. Our work in chapter 3 takes a similar approach to attention-based spatial reasoning. However, we emphasise environmental properties to inform general navigation strategy rather than specific objects within the scene.

2.4 Summary

This chapter presented overviews of the current state-of-the-art techniques in the fields of reinforcement learning and imitation learning, disentanglement in VAEs, subgoal search, multi-task learning, and visual navigation. The remainder of this thesis will attempt to combine these fields into a multi-task capable, reinforcement and imitation learning enabled, hierarchical robotic system.

Each of the fields mentioned above has some limitations which can be addressed by combinations with other fields' techniques and ideas. In this thesis, we present a series of contributions which address these limitations and attempt to further the progress towards a more applicable AI system for robot deployment. In the next chapter, we will first prove the validity of our robotic system where the AI agent commands low-level operation functions to achieve traditional robotic tasks. More specifically, we will enable a reinforcement learning agent with multi-task learning capability to perform visual navigation tasks in multiple environments. In Chapter4, we will present the work where we attempt to disentangle the learned latent space into skill and knowledge subdomain following the PD memory paradigm. We show that the learned skill and knowledge in multi-task learning agents can be disentangled, which will enable the generalization of unseen tasks, and improve the efficiency of information sharing. In Chapter5, We utilized the idea of sub-goal to solve long and complex tasks with imitation learning multi-task capable agents. Together, these contributions will collaborate and drive an autonomous AI robotic system, capable of solving complex tasks and lay the foundation for future work.

Chapter 3

Robot in a China Shop: Using Reinforcement Learning for Location-Specific Navigation Behaviour

3.1 Problem Definition

As technology advances, the potential applications of modern robotics become more diverse. We are assigning more tasks to robots in more places and exposing them to more difficult challenges. No longer fixed in the assembly line repeating the same motion, robotics has progressed from vacuuming a room to delivering take-out food and bartending, from working the assembly line to the management of an entire factory. As their functionality expands, so does the variety of possible working environments. Currently, Robots are often designed to operate in a single type of working environment. The working environment is often known, any interactions that may occur within this workspace are pre-defined and known by the robot, and a human operator is ready to step should anything happens. A larger space, such as a house, can be treated as a single environment (although this is an oversimplification) and dealt with as such. However, this is no longer possible when the scale of the working environment of a robot could encompass a

large building or an entire city. This environmental limitation should be addressed as soon as possible.

In the field of reinforcement learning, it is natural to look to human behaviour for guidance in order to free robots from the constraints of a single environment. When we enter a new environment, we frequently behave differently, and this behaviour change is not caused by the task at hand, but rather by the change in environment. When we walk into a china shop, we walk gently and carefully to avoid any collision, as opposed to walking down an empty hallway, where we don't need to worry about breaking anything and speed through. Robots could utilize this ability by comprehending their surroundings and adapting their behaviour accordingly.

Furthermore, as humans, we, compartmentalise our knowledge and memory so that we can effectively work with only a subset of them rather than all of them at once. We don't need to remember the city map when we go to the grocery store, even though both are important navigational information. This concept serves as additional motivation for our multi-task learning approach.

In this chapter, we define the combination of these abilities to be the problem of multi-environment navigation: A robot, controlled by a single artificial intelligence model, should have the ability to detect and navigate through several different types of environments. In each other kind of environment, the artificial intelligence model will use a strategy fit for the type of environment it is operating in. To solve this problem, we focus on the field of visual navigation, as visual input information (relative to other sensory inputs) will be most effective in understanding the difference between different environment settings. In particular, we focus on navigation from visual sensors and propose a new network architecture: Multi-environment Reinforcement-Learning in Navigation (MERLIN). However, the proposed techniques could also be applied to input data from other sensing modalities such as lidar and sonar.

Visual navigation is an active field of study, however, most of the previous work concentrates on enhancing navigation accuracy within a singular environment. In this study, we present a hierarchical reinforcement learning architecture where visual navigation in multiple environments can be achieved. This architecture deploys a Siamese network feature extractor and numerous expert networks with attentive gating, paired with a particular classification branch to encourage the network to recognise the difference between surroundings. We find our model outperforms

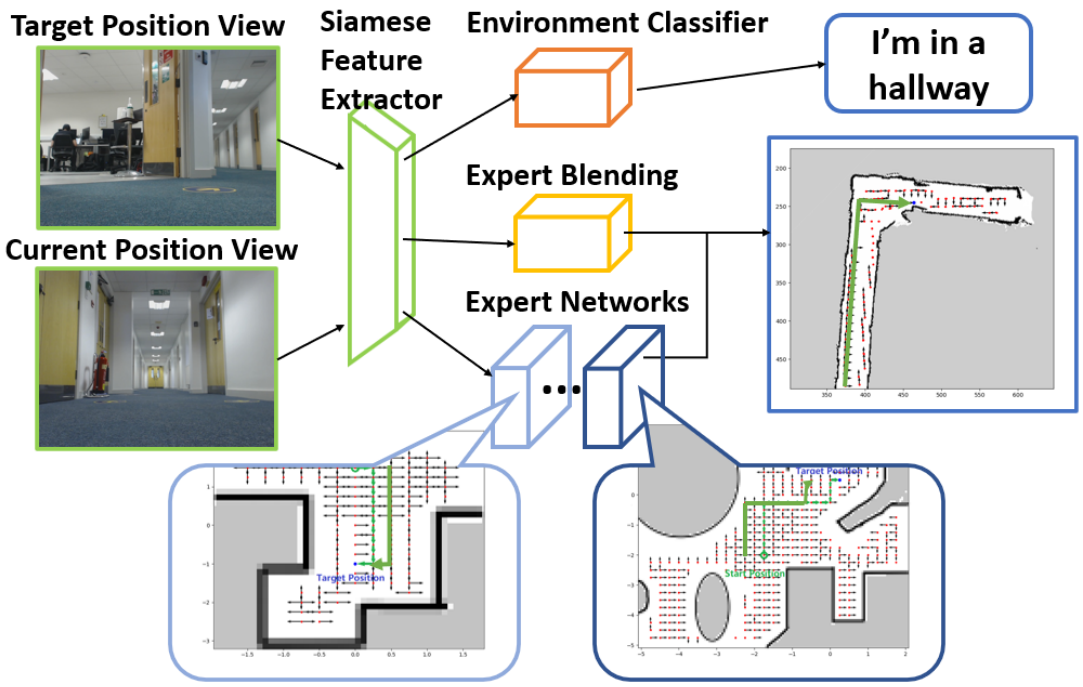


Figure 3.1: The target position view and current position view are given to the agent, the environment classifier encourages the network to distinguish between environments in earlier layers, while multiple expert networks' outputs are blended to produce the final policy.

the state-of-the-art visual navigation models. Furthermore, by encouraging environment classification, we are able to achieve better results in learning performance, accelerating learning in multi-environment navigation.

In summary, we define a the problem of multi-environment navigation and propose a new model for approaching this problem. The main contributions of this study are as follows:

1. Define the problem of multi-environment navigation.
2. Propose a new multi-task learning approach to visual navigation.
3. Propose the inclusion of an intermediate environment classification loss to accelerate learning.

A preliminary version of this work was presented in ICRA 2021 [114]. This extended version greatly enhances the environmental recognition approach and evaluation. In the original

manuscript, the network identified a unique label for each individual environment. This limited information sharing between environments. In this manuscript the environment classifier recognizes a broad set of overlapping environmental properties. Each environment now has a variable number of associated property labels, and multiple environments share the same labels. This helps provide explicit guidance about the similarities of different environments to improve the skill transfer in multi-task learning. We also doubled the number of environments used in the evaluation benchmark compared to the original publication and showed improved performance over them all.

The rest of this chapter is organised as follows: Section 3.2 introduces our methodology, including the additionally expanded environment property classifier. In Section 3.3 we evaluate our system both quantitatively and qualitatively, and we conclude in Section 3.4.

3.2 Methodology

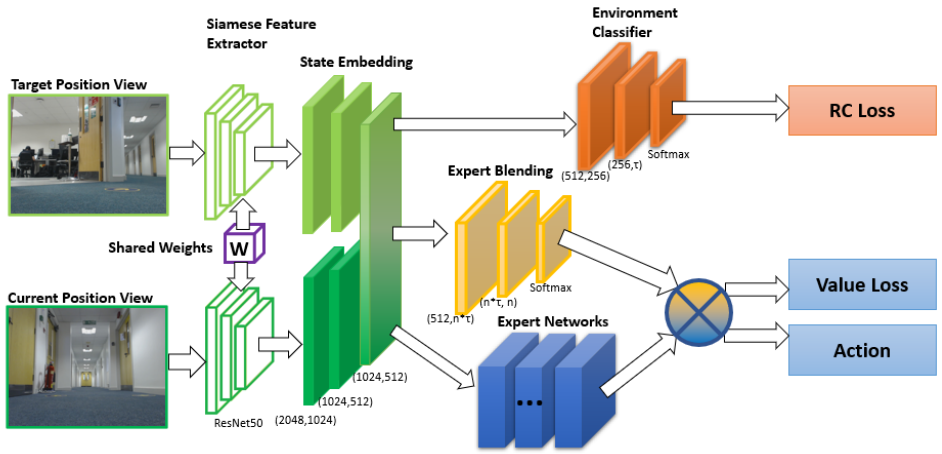


Figure 3.2: Our architecture can be divided into 4 major components: A Siamese feature extractor (green), n sub-expert-networks (blue), an attention network (yellow), and an environment classifier network (orange).

We start from the problem of AI-based multi-environment visual navigation. We define visual navigation here as using only the visual input from the front camera of the robot and an image from the goal position as the input for the system. The objective of the agent is to be able to navigate in multiple types of environments, learn new environments without forgetting learned

ones and be able to generalize to unseen environments. The agent can navigate the environment through actions: step forward, step backwards, turn right and turn left (90 degrees). Given a target, the agent will only be given its current observation Current Observation (I_c) and a view of the target Target Observation (I_t) . A single agent should be capable of solving a set of similar navigation tasks placed in different environments by training a policy $\pi(a_t|s_t, \tau)$ and value function $V(s_t, \tau)$ for each task, while maximizing the reward for each task in the task set $\tau \in \mathcal{T}$.

To tackle the specific problem of multi-environment/task visual navigation, we utilize the idea of multi-task learning with a reinforcement learning framework, and we propose a new architecture: MERLIN. Our architecture can be largely divided into 4 major components as shown in figure 3.2: A Siamese feature extractor, n sub-networks, an attention network and an environment classifier network. In the following section, each of these components will be described in turn.

3.2.1 Siamese Feature Extractor and Joint State Embedding

The network has 2 branches joined at the input to form a Siamese network as shown in figure 3.3. The expected functionality of the Siamese feature extractor is to capture the input state through feature extractors as well as to identify commonly useful information for all tasks and the feature characteristics to identify different tasks.

The Siamese feature extractor takes input in the form of a vector of the 2 observation images: the current agent observation I_c and the current target I_t for $I \in \mathbb{R}$. Each image will be fed into a different branch of the Siamese feature extractor network. Both the target state input and the current state input are first put through convolutional feature extractors F_e. The feature extractors share the same weights ω_S, between branches, and the output of each branch will then go through a normalization function F_n before concatenating into a state embedding. For the current state observation $O_C = (I_T, I_C)$ that contains the feature information from both input images, the feature embedding $\mathcal{F}$ for state s is defined as:

$$\mathcal{F}(s) = \{F_n(F_e(I_t|\omega_s)), F_n(F_e(I_c|\omega_s))\} \tag{3.1}$$

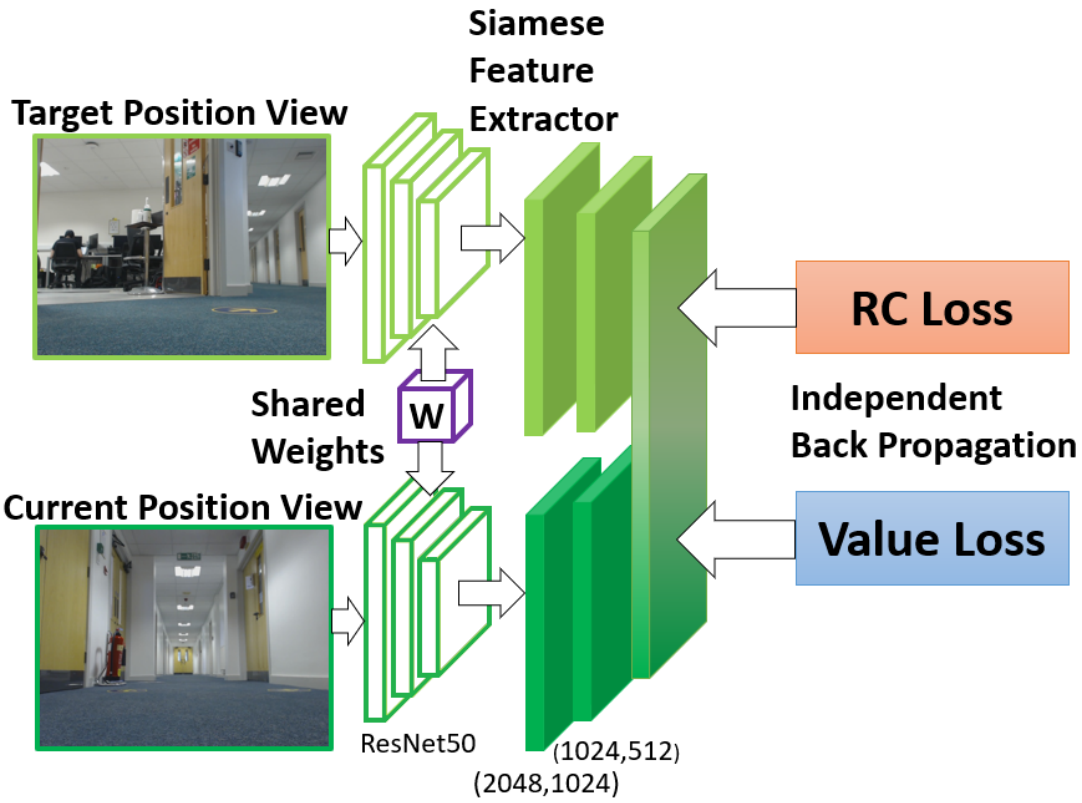


Figure 3.3: The shared layer is a Siamese network with input heads for both the target state view and the current state view. The combined state embedding concatenates information from both observations.

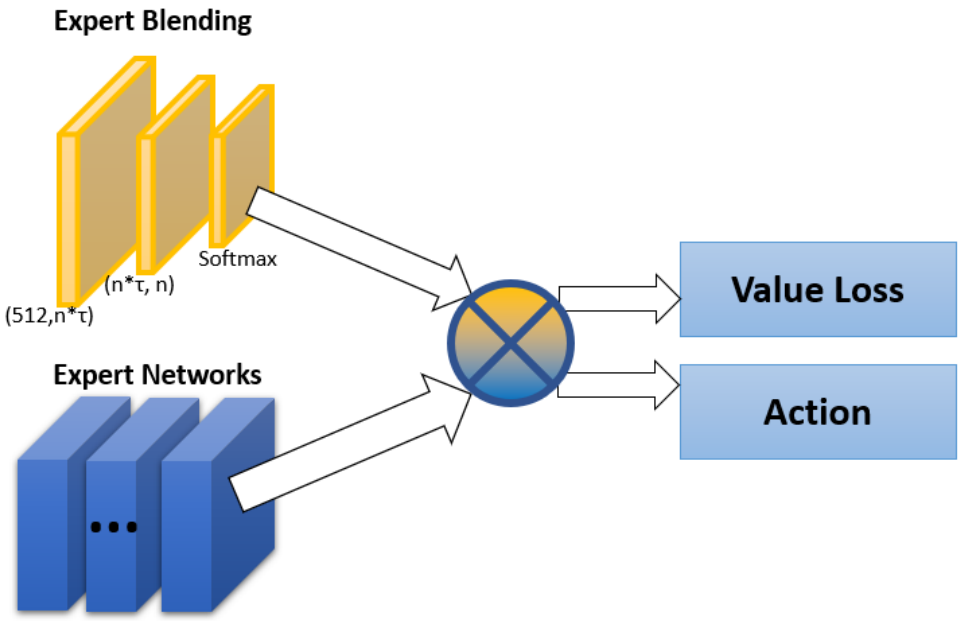


Figure 3.4: Each expert sub-network produces a policy function, the expert blending network assigns weights to each sub-network, the weighted sum is then used to produce the agent's action and the value loss.

The Siamese feature extractors are updated by losses flowing through both the RC network branch and the sub-network/attentive network branch. This means the features are required to simultaneously be effective at solving the RL navigation task, and capable of distinguishing different categories of the environment.

The Siamese feature extractor is trained separately from the rest of the network to save time. This could cause a discontinuous gradient over the input latent. However, in the current framework, the visual input changes dramatically when the agent performs a turning action, a discontinuity is expected, and the network should learn to cope with this.

3.2.2 Task-Specific Expert Policy Sub-Networks

The Number of Sub-Networks (n) serve as the expert networks that learn the knowledge and skill to solve a specific task. The value of n is determined by a range of factors including the number of tasks, the similarity between each task, the similarity between each environment, etc.

The number of n could be determined by the amount of information required to learn in the training dataset. However, there's no easy way to calculate the learned information contained in these sub-networks. Therefore in this work, the number n is largely proportional to the number of environments. This number of sub-networks is affected by a number of factors such as the current task, the environment type, and the size of the expert networks, etc. Currently, this number is determined experimentally to have the optimal performance in the number of e we tested. There exist a trade-off between network expressiveness and computational expense. Excess sub-network will ensure the information learned during training is fully preserved but at the cost of computation consumption. The optimizer will learn to ignore excess sub-networks when given more resources than required as shown in the result of Bram [17].

The specific architecture of the sub-networks can be altered to fit the scenario. It is possible to have a variety of different expert networks that would work better for different tasks or task settings combined within the same agent. As we are working on the subject of visual navigation, the sub-network architecture in this study is designed for visual information processing and navigation tasks: A softmax layer maps the last hidden layer of each network to an Action (a) dimensional vector to produce Action Probabilities (A_i) and a linear layer outputs the value function V_i for each expert network with a specific policy π_i:

$$A_i = softmax\left(\pi_i(E_i(\mathcal{F}(s)|\omega_i)|\omega_A)\right) \tag{3.2}$$

And the corresponding value function can be computed for each expert network:

$$V_i = E_i(\mathcal{F}(s)|\omega_i) \cdot \omega_V \tag{3.3}$$

Where E represents the RL network used for the actor-critic branch, and ω_V is the overall weight for the value function. If the action space is different between tasks, the size of A should be the largest action space size of all tasks.

3.2.3 Attentive Task Allocation and Soft Blending Network

In our framework, the attentive task allocation network select and synthesise output from task-specific expert sub-networks. By assessing the relevance between each sub-network and the current environment, the allocation network will assign an weight to each sub-network's output. The weighted-sum is used to determine the final action decision.

The attentive task allocation network first takes the output features $\mathcal{F}(s)$ from the feature extractors and recognizes the corresponding expertise required by different tasks. While each expert sub-network produces a policy function, the attentive network assigns a distribution weight ω_τ to these policy functions according to the estimated relevance of expertise. Therefore, the attention weight for the expert sub-network τ can be computed as:

$$\omega_\tau = softmax\left(\{Att(\mathcal{F}(s)|\omega_{Att_i})|i \in \{0..n\}\}\right) \tag{3.4}$$

The softmax normalizes the weights ω_{Att_i} to sum to 1. By concatenating the action distribution outputs of each expert network into a vector A. The final policy is then based on the dot product of the attention weights against the experts' action distributions:

$$\pi(\alpha|s) = \sum_{i=0}^{n} \omega_{\tau_i} \cdot A_i \tag{3.5}$$

where ω_{τ_i} is the ω_τ for the ith expert network.

This policy will determine the action taken by the agent during each timestep. The Attentive Distribution Weight (ω_τ) is also used to compute a value function for reinforcement learning branches of the network. By taking the dot product of the expert value loss from the sub-networks with the attention vector, the final value loss can be computed. For the expert values $V_\tau = V_0, V_1 ... V_n$, the combined value function is:

$$V_r(s) = \sum_{i=0}^{n} \omega_{\tau_i} \cdot V_i(s) \tag{3.6}$$

This value loss is used to update the expert networks, the attentive task allocation network, and the Siamese feature extractor, but not the RC networks.

3.2.4 Environment Classifier

During our experimentation in the original conference publication [114], we found that the optimal number of expert networks did not necessarily correspond to the number of environments, and was often lower than the number of environments. This led us to theorize that the visual navigation task involves some shared skills that can be transferred between environments. Skills such as obstacle avoidance and path planning are likely shared across different environments during learning, which would cause common behaviours across similar environments and a reduced amount of expert network resources.

To better understand the role of environmental properties in the navigation task, we introduced an environment classifier (EC) that recognizes labels corresponding to properties of the environment itself. These labels describe the different aspects of the scene. For our experiments we prepared two sets of labels, the "room type" (RT) label which uses terms such as bathroom, living room, etc. to describe the functionality or purpose of the environment. Another set of labels describe the different features of the room (RD) such as size, layout, and obstacles present in each scene. Hence, we define $\mathcal{M}_\tau$ as a vector which contains descriptive classification labels of the environment, and define the loss function of the environment classifier as an MSE loss:

$$L_{rc}(s) = |\mathcal{RC}(F(s)|\Omega_{rc}) - M_\tau| \tag{3.7}$$

As the environment classifier has a much shallower network depth, the classifier loss L_{rc} could have an overly large impact on the shared network, therefore a balancing coefficient is applied to the loss. For similar reasons, the value loss is also applied with a coefficient. The final loss of the end-to-end system is:

$$L = \alpha * L_{value} + L_{action} + \beta * L_{rc} \tag{3.8}$$

3.3 Evaluation

To evaluate our approach, we experimented with our agent's ability to perform visual navigation tasks in both simulated and real environments. The agent will be trained in multiple different themed environments simultaneously. In different themed environments, the agent is expected to recognize which environment it is in and perform navigation tasks in an effective manner. Through the use of differently weighted expert networks, the agent selects specialist expert networks and blends their policies. This will train the agent to be able to handle multi-tasking, as well as recognize different tasks. The RL algorithm used in the experiments is a multi-thread A3C. The network backbone for the sub-networks is the same as the SOTA model [124]. We let $n = \lfloor(|e| + 2)/4\rfloor + 1$ where e is the number of environments present in the training set. The symbol $\lfloor\rfloor$ defines floor operation. The reward is inversely related to the length of the path taken by the agent to reach the target position: Negative rewards accumulate with a time penalty (p_{stp}). The path length l is penalised at a rate of $-r_{stp}$ per step ($p_{stp} = -r_{stp} * l$). The agent

also takes an additional penalty $p_{cr} = -r_{cr} * n_{cr}$ for the number of times n_{cr} when the agent hits obstacles. Each episode is limited to l_{lm} timesteps to deter reward hacking and avoid the agent getting stuck. Reaching the target position before this limit will result in a positive reward r_{ter} for x points of reward.

$$r_{ter} = \begin{cases} x & \text{for} \quad l \leq l_{lm} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Exceeding the time limit will result in ending the episode and a negative reward p_{ter}.

$$p_{ter} = \begin{cases} -x & \text{for} \quad l > l_{lm} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The final reward R is given by the sum of all rewards and penalties:

$$R = p_{stp} + p_{cr} + r_{ter} + p_{ter} \tag{3.11}$$

The agent will receive an image of the target position I_t and the current position's view I_c. In the simulation, a small amount of random noise is added to the current position before sampling I_c to ensure generalization. Our agent is implemented with Pytorch and trained on Nvidia Geforce GPU servers. The evaluation is done in several experiments both in simulation with 5 different random seeds, and in the real world. The average performance is recorded.

3.3.1 Experiments

We prepared both simulated and real-world environments for the experiments as shown in table 3.1. To create the simulated environments, we used the 3D simulation environment AI2-THOR[55]. AI2-THOR is a 3D simulation program used for machine learning. The simulated environment consists of themed rooms such as living room, bathroom, kitchen, etc. We created the regularly sampled environments by allowing the agent to move forward and backwards as well as turning 90 degrees on a square grid. The agent will be dropped randomly into the environment and given a random target which is reachable from the starting position. The square grid has a grid size of α with each position g_i being a sample from SO(2) (i.e. comprising of x, y, θ). The square grid G has a grid size of α with $g_i \in G$ being a sample from SO(2) (i.e. comprising of x, y, θ). Let $G(x, y, \theta)$ denote the view I obtained at a certain location and

Environment Name	Discription	Size	Space Labels
Env1	Simulated Kitchen	Medium	Open, Full
Env2	Simulated Living Room	Large	Narrow, Full
Env3	Simulated Bathroom	Small	Open, Empty
Env4	Simulated Small Bathroom	Small	Open, Empty
Env6	Real Common Room	Medium	Narrow, Full
Env7	Real Corridor	Large	Narrow, Empty

Table 3.1: Simulated and real training environments are used in the experiments.

orientation in the grid. The target is described through the view of the agent I_t at the target position $g_T = G(x_T, y_T, \theta_T)$. The current position $g_C = G(x_C, y_C, \theta_C)$ of the agent is given through a view of the agent at its current position I_c. To ensure the generalization capability of the agent and simulate the navigation error of an actual robot, the current view is sampled randomly near each grid point. The random sample position is selected by adding Gaussian noise proportional to the grid size $\Delta g \sim \mathcal{N}(0,\ (\alpha * \rho)^2)$ to the x and y coordinate of the original state position. The sampling position $(\hat{x}, \hat{y}, \theta)$ can be formalized as $\hat{x} = x + \Delta x \sim \mathcal{N}(x,\ (\alpha * \rho)^2)$ and similarly, $\hat{y} \sim \mathcal{N}(y,\ (\alpha * \rho)^2)$ The rotation of the agent is unchanged for the sampling. Resulting in the view at $g_{sample} = G(\hat{x}, \hat{y}, \theta)$ as the actual input to the agent.

The agent will then try to find the shortest path to reach the target position. In our experiments, 4 environments are trained simultaneously, each using the same amount of computing resources, the average episode length and percentage of successful runs across all 4 environments are used to evaluate their performance. In the second part of the experiments, we changed the random sampling method of the current view to uniform random sampling. ($\Delta g \sim \mathcal{U}(0,\ (\alpha * \rho))$) This will introduce greater spread in sampling and increase the difficulty for evaluating the agent's ability to generalise. A uniform random sampling noise $\mathcal{U}(0,\ (\alpha * \rho))$ with respect to grid size is added to the actual current position, and the current view is sampled from the new position.

To train the agent in real-world environments, we collected real-world data using a Turtlebot from various locations with different themes such as a hallway, living room, common room, etc. as shown in figure 3.5. As the robot already has drifting errors, there's no random sampling used

Figure 3.5: Example images from the real-world environments dataset

in the real-world datasets. Using these images, we produced gridded real-world environments similar to the simulated environments which can be trained offline. A total of 3 different real-world environments were used for training: A residential living room, a university common room, and an office corridor. They represent 3 types of spaces: a small enclosed space with few obstacles, a large open space with many obstacles, and a narrow enclosed space without obstacles. The agent is expected to behave differently in each of these types of spaces.

In the last section of the experiments, we improve on this and demonstrate our model directly on a real robot. To achieve this, a Turtlebot3 (shown in figure 3.6) is used to navigate in the previously trained environments. The turtlebot3 is capable of navigation using lidar and odometry, the localization function is used to determine the current position of the robot. The robot is fitted with an additional camera in the front to provide image input. During the experiment, the robot will be given an image of the target and captures a live image from its camera as the current view. The images will be used as input for a model previously trained on the off-line dataset.

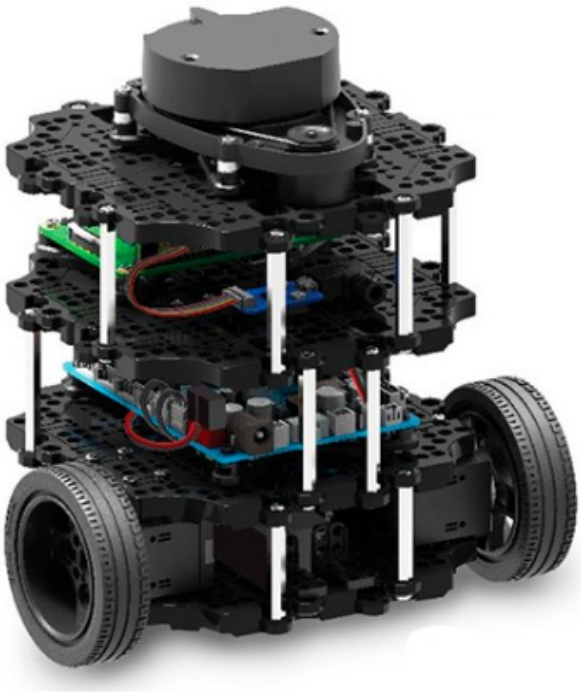


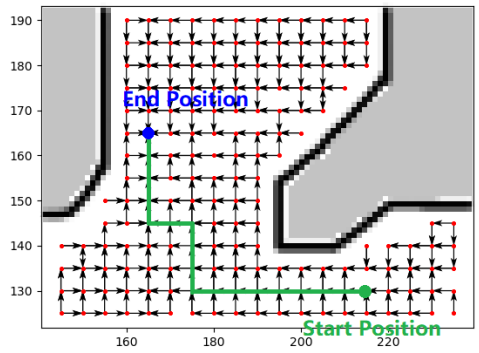
Figure 3.6: A typical Turtlebot3, the robot used for the live experiments has an additional camera mounted to provide the front view image.

The robot will attempt to perform the instruction and move to the next position and repeat until it reaches the target position. [1]

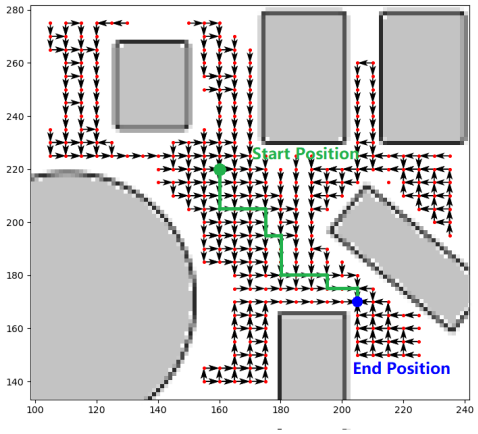
3.3.2 Baseline comparison

We first compare our agent with the SOTA target-driven navigation model [124] and perform an ablation study of our technique. The SOTA model uses a network structure with generic Siamese layers and scene-specific layers. However, the scene-specific layers are trained specifically for each environment, resulting in expert performance in a single training environment, but an inability to share learning between different environments. The SOTA model is trained in different environments both separately (referred to as Expert1 to Expert4) and concurrently (referred to as "Joint Expert"), where the SOTA model is expanded to have the same latent

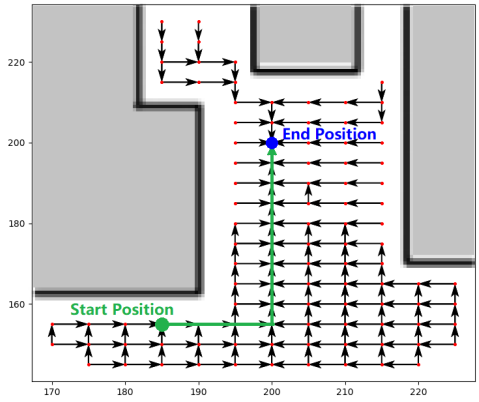
[1] A video showing this experiment can be found at: https://youtu.be/ayjwNdCOkbw.



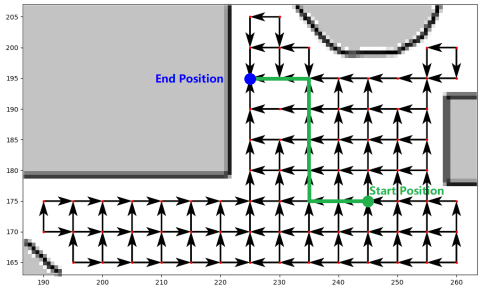
(a) Env1: simulated kitchen



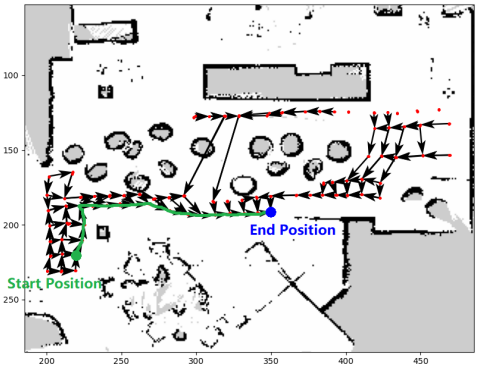
(b) Env2: simulated living room



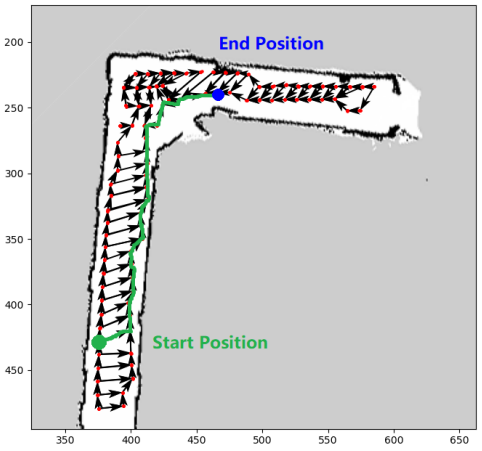
(c) Env3: simulated bathroom



(d) Env4: simulated tiny bathroom



(e) Env6: real common room



(f) Env7: real corridor

Figure 3.7: The agent makes large strides before turning in open spaces and avoiding walls when in narrow spaces.

Sim Dataset	Env1	Env2	Env3	Env4	Ep. Length	Avg. Reach Goal (%)	RC accurcy (%)	Converge Step
MERLIN	**22.65**	**27.20**	**22.23**	**20.31**	**24.33**	**93.40**	83.80	23K
Expert1	23.48	199.61	198.82	198.03	154.81	24.00	-	6K
Expert2	198.04	42.54	196.89	195.70	158.13	23.30	-	10K
Expert3	199.22	199.21	33.95	196.90	157.15	22.65	-	5K
Expert4	199.22	199.21	197.68	19.19	153.69	24.23	-	7K
Joint Expert	25.38	43.69	23.50	22.58	28.73	91.65	-	31K

Table 3.2: Navigation task step count and success percentage in the simulated dataset.

Real Dataset	Env1	Env2	Env3	Ep. Length	Avg. Reach Goal (%)	RC accurcy (%)	Converge Step
MERLIN	**8.50**	44.28	**40.30**	**30.19**	**90.63**	82.40	19K
Expert1	11.20	197.36	199.40	135.92	33.42	-	4K
Expert2	197.03	**27.83**	197.61	140.71	31.85	-	20K
Expert3	199.40	199.11	45.24	144.53	30.12	-	6K
Joint Expert	49.72	69.75	42.85	50.73	80.90	-	25K

Table 3.3: Navigation task step count and success percentage in the real-world dataset.

capacity as MERLIN for fair comparison. Each model is trained with 5 different random seed, and their average performance is recorded. All models are trained until they converge to over 99% success rate, the average episode length and the number of time steps taken to converge are recorded.

As shown in both Table 3.2 and 3.3, the separately trained SOTA models show an inability to perform navigation task in any environment other than the one it was last trained on. The jointly trained baseline can complete the tasks in all different environments, but it requires a 30% longer training time and has a lower performance than our proposed technique. Additionally, our agent is able to outperform the specialist experts in most of their corresponding environments. This suggests that there is a sharing of expertise between environments, which can take advantage of our expertise-blending approach. The simulated kitchen, as shown in figure 3.7b and table 3.2, is a larger environment with more grid points compared to other environments. The joint expert performs more poorly in this environment compared to its performance in the rest of the environments. A similar pattern can be observed in the real-world results as shown in table 3.3, the joint expert has particularly low performance in the largest environment Env7. The joint expert also shows a lower performance across all tasks and requires longer training to converge. It appears that the improvements offered by the proposed approach scale with the number and

the size of the environments. The MERLIN model also has a 24%-26% faster converge speed. This indicates that when the number of learnable parameters increases, it is possible that our approach will still be able to converge on more difficult tasks when the joint expert cannot.

3.3.3 Ablation Study

We also performed additional experiments where we increased the number of environments training simultaneously to 8 different environments (8 Envs model). We compare this model's performance against the model trained in 4 environments (4 Envs model). The model trained in 8 different environments is tested on the same four simulated environments for a fair comparison.

MERLIN	Ep. Length	Avg. Reach Goal(%)	RC Accuracy(%)	Converge Step
4 Envs	**24.33**	93.40	83.80	23K
8 Envs	29.92	**96.70**	100	73K

Table 3.4: Performance comparison with agents trained in different numbers of environments.

As shown in table 3.4, the performance of the agent increased with an increasing number of environments during training. While the average percentage of reaching the goal within a time step limit and the room classifier recovery accuracy are about the same, the average episode length is not a reliable metric as each environment is different in size. Interestingly, the number of training steps used to reach convergence for the 8 Envs model is higher than the direct proportion of the converge step of the 4 Envs model. Although the number of environments doubled, the converge time is about 3 times greater. We suspect this is caused by the increased number of expert sub-networks to handle the increasing amount of information and more complex attention mapping.

We also compare against the simplified room classification approach from our original conference publication. We train the original MERLIN agent with and without the RC branch, as well as replace the RC branch with RT and RD branches. The results are shown in table 3.5.

As shown in the table, the Joint Expert over-fitted to the training set quickly, unable to adapt to the random route in the testing set. Without the RC branch (MERLIN-RC), the multiple expert networks are still able to learn the navigation task, however, its performance is lower

Model	Ep. Length	Avg. Reach Goal(%)	RC Accuracy(%)	Converge Step
Joint Expert	438.9	11.74	-	**41K**
MERLIN-RC	41.99	93.62	-	120K
MERLIN	29.92	96.7	100	73K
MERLIN+RT	23.42	96.51	100	100K
MERLIN+RD	**15.21**	**97.15**	100	230K

Table 3.5: Ablation study on RC branch

than a normal MERLIN network. When we improve the RC branch with a room type label (MERLIN+RT) and room description label (MERLIN+RD), there is a significant improvement in performance with the latter change. We suspect this is because the room type labels only identify the environment's functionality, which is only weakly correlated to navigation-relevant properties such as layout and size. The room description label on the other hand gives a more detailed description of each individual environment, including its layout, size, and occupancy. These are all very helpful for informing the navigation task, and for transferring shared knowledge between different environments.

3.3.4 Qualitative Multi-environment behaviours

In figure 3.7, we provide examples of the behaviours of the agent in different environments. The agent has different behaviour when dropped to a random position in each environment. The vector fields are formed by examining the trajectories from all possible starting positions to the target, and the green trajectory shows one complete example trajectory. In open spaces such as the simulated living room 3.7b and the simulated kitchen3.7a, the agent tends to make strides and turns for localization. In narrow spaces such as the simulated bathroom 3.7c the agent would prioritize moving away from walls. In the real corridor 3.7f the agent would have much fewer turning actions during the narrow hallway but shows turning behaviours in the middle section where the space is relatively open.

Our qualitative experiments have shown that different environments can cause the agent to behave differently for optimal performance. In particular, we have observed that certain common

behaviors can be observed in scenes with corresponding characteristics. For example, the layout of the room can determine the movement strategy of the agent, either making long strides for speed or staying in the center to avoid any collision. Similarly, the size of the scene can determine the size of the strides the agent would make in one direction. Finally, the clustering of objects and the functionality in the scene will help determine how "careful" an agent should be in the scene. We hypothesize that these descriptive labels will enforce semantic learning in the early layers of the network, and improve the performance when learning multiple scenes. With the RC branch, the network should be able to learn common skills shared across different environments. thereby improving its capability in transferring knowledge from one environment to another, and generalization across different environments.

3.3.5 Generalization and Noise Resilience

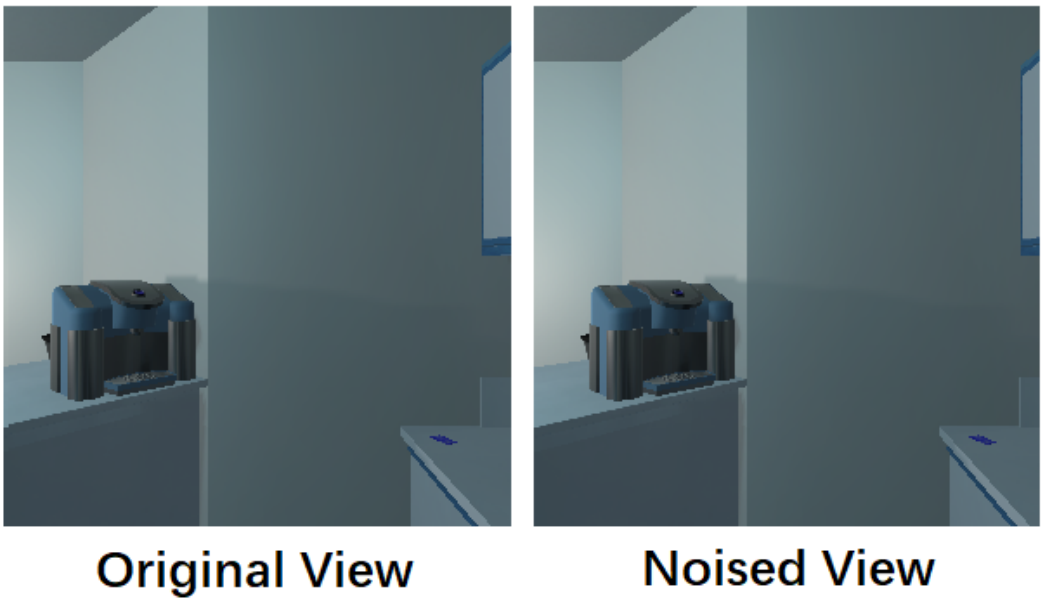


Figure 3.8: The original view and position compare to the noised view and position at a 10% noise level. The agent's position shifted within 10% of the unit distance between each step position.

In the third experiment, the time limit is tightened for completing the episode, and the noise ratio for current view sampling is increased. The sampling method is also changed from Gaussian noise to a uniform noise scaling with the grid size to increase difficulty. A sample size of 50

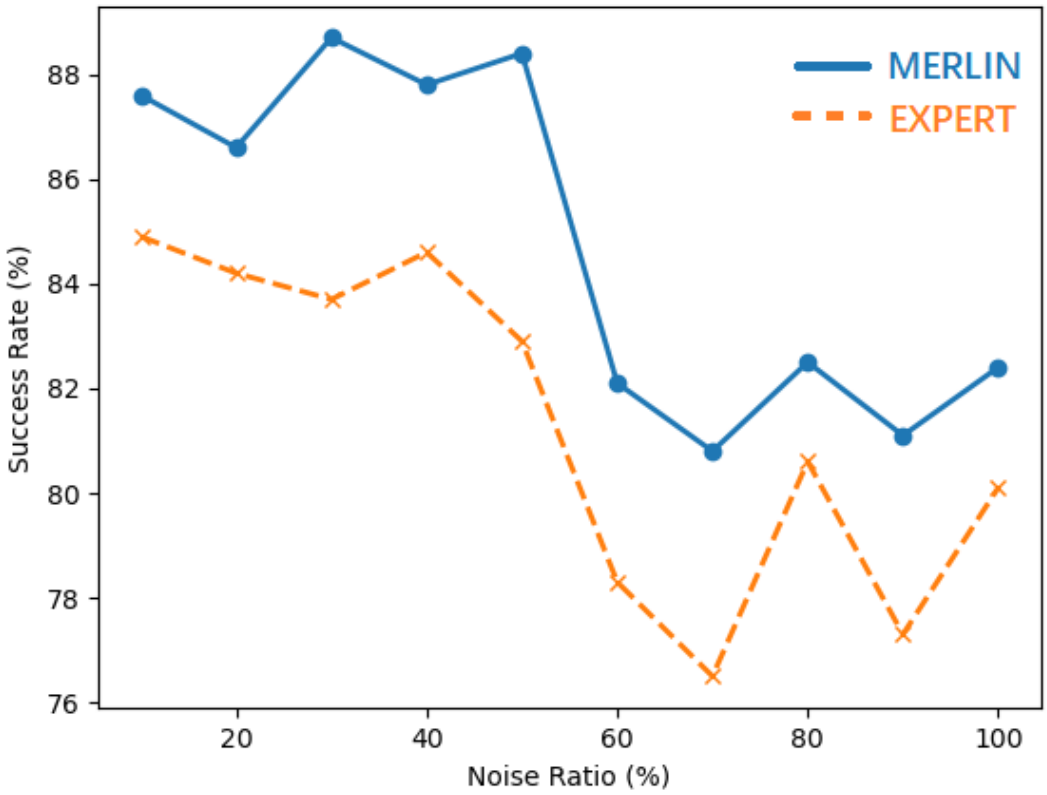


Figure 3.9: Success Rate drops with increasing noise ratio in current view sampling. The blue line indicates MERLIN's performance, and the orange line indicates the joint expert.

is used for each noise level, and the agent will perform 1000 episodes across the 4 simulated environments. As shown in figure 3.9, MERLIN outperforms the joint expert consistently and maintains a more than 80% success rate until the noise level reaches 100% of the grid size. A significant drop in performance occurs at around 50% noise level. This is due to the possible sampling positions of each grid starting to overlap with each other. At 100% noise level, the sampling position will have a low chance of drifting to another state's position, confusing the agent. These results indicate the MERLIN agent has a good generalization ability within each environment and is not overfitting to the training environment. This also indicates that an agent trained offline in a gridded version of a real-world dataset could potentially be transferred to operate in the real world.

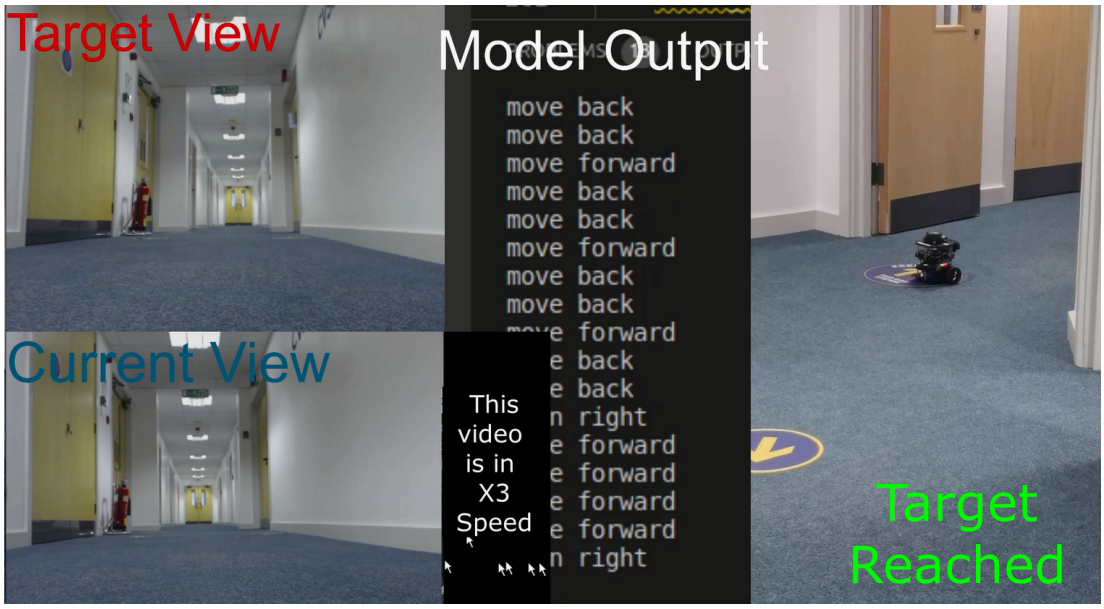


Figure 3.10: Live demonstration shown in the video clip, the robot successfully reached the target position following action comments from the agent trained in simulation.

3.3.6 Live Demonstration

In the last experiment, the live demo shows the Turtlebot performing a target-driven navigation task in the real-world location of Env7 (real corridor) using the MERLIN model, shown in figure 3.10. [2] The robot successfully completed the task with only a few missteps outside the optimal path, likely caused by lagging and drifting errors. We can also observe the robot making a straight line in the hallway, but turning and making strides in the relatively open area in the middle. Compared to the gridded simulated environments, the robot spends more time turning for localization and attempting to correct drift errors. As the model is trained in discrete environments, it has a natural deficiency in handling inaccurate turning angles, this could potentially be improved by generalizing over-rotation during training. The model's solution to this is to keep turning until it finds a recognizable direction. However, this strategy has difficulties with excessive lagging and drifting errors.

[2]The video can be find in the following link:https://youtu.be/ayjwNdCOkbw

3.4 Conclusion

In conclusion, we introduced the multi-environment navigation problem in the field of robotic navigation and proposed a multi-task deep reinforcement learning framework to approach this problem through visual navigation. While SOTA studies in either multi-task learning or visual navigation address some aspects of this problem, this study combines both fields and takes advantage of both sides to formulate an improved solution. Our approach utilized multi-task learning and reinforcement learning, with the attention task allocation and environment type classifier, which has enabled the agent to navigate in multiple environments with one training session. Overall, the MERLIN model outperforms the SOTA model in the multi-environment target-driven navigation tasks in both performance and training speed. Interestingly, MERLIN also outperforms specialist single-environment expert networks even in their own training environment. We observed different behaviours depending on the surroundings in both the simulated environments and real-world environments. We also demonstrated the model's ability in operating in the real world even when trained offline in discrete environments. We have proven our model's capability in navigating through multiple different environments efficiently while recognizing they are different and employing mixtures of expert networks accordingly.

It is foreseeable that upcoming robots will require more multi-tasking capability than navigation in multiple different environments. They may also need adaptive skills for undertaking various non-navigation tasks in a variety of locations. Mimicking human's ability to adapt to environments is going to be vital for robots and provides a great challenge for the field of robotics and artificial intelligence.

During the experiments, we find the number of sub-networks is not directly proportional to the number of environments undergoing training, the optimal number of sub-networks is often less than the number of environments. We suspect this is caused by the transferring of skill and knowledge between each navigation task, as similar behaviour can be observed across similar types of environment. To better understand this phenomenon, the next Chapter will attempt to disentangle the information learned during multi-task learning, and explore the transferring of knowledge and skill between each task in the learning process.

Chapter 4

SKILL-IL: Disentangling Skill and Knowledge in Multitask Imitation Learning

4.1 Problem Definition

In the previous chapter, we explored the application of multi-task learning in combination with reinforcement learning in the field of robotic navigation. We concluded that the multi-task learning techniques provided an enhanced navigation strategy by considering the type of environment. In this chapter we will seek to learn more about the sharing and transferring of information within such a framework. Importantly we will explore more fundamental and flexible ways to apply multi-task learning to a broad variety of tasks beyond navigation.

To achieve this, we must learn to generalize and share information across different domains and re-combine this information for unseen tasks. Researchers struggle to transfer expertise efficiently between tasks, or even between sub-problems of the same task. Meanwhile, research on transferring to previously unseen tasks (zero-shot RL/IL) growing in popularity. To approach this problem, we find inspiration from human learning behaviours. We as humans spend years learning varied tasks, from how to walk and talk, to writing papers or gymnastics. This learning is a process of imprinting memories in our brain, not entirely dissimilar to training the weights

of a neural network. Thanks to recent developments in neuroscience, we know our memories fall into two categories, representing either procedural memories or declarative memories. With both procedural knowledge and declarative knowledge, we can navigate this world and complete tasks. Procedural Memory, or "**Skill**" is the memory required for the agent to perform a certain task in general [71, 86]. Declarative Memory, or "**Knowledge**", involves memory specific to the environment the agent is operating in [71, 19]. For example, when we are driving to work, this requires us to have the **skill** of driving a car (procedural memories), and the **knowledge** of the route to get to work (declarative memories).

Most tasks require both skill and knowledge simultaneously to complete. However, these are independent and transferable. We can also use our driving skills to drive to the store, or we can use our knowledge of our workplace to cycle to work. Neither of these transfers would imply additional training. This capability would be invaluable in multi-task learning, as each problem requires a different combination of knowledge and skill. Generally, every possible combination of knowledge and skill is treated as a separate learning problem, or every skill is trained independently to generalize over all knowledge. This greatly increases the difficulty of multi-task learning, leading to scalability issues and unrealistic training requirements. In this work, we propose SKILL-IL as a new approach to multi-task IL which explicitly disentangles and shares both skills and knowledge across tasks, as shown in figure 4.1.

This work is inspired by psychological studies with amnesiac patients and non-amnesiac patients [111]. These studies have shown that these types of memory are not fully entangled in human learning. It is possible to separate procedural memory from declarative knowledge in the learning process. To perform any task, an agent requires both procedural knowledge and declarative knowledge. The disentanglement of these two types of knowledge would benefit any multi-task learning model as skills should be transferred between the same tasks in different environments, and knowledge should be transferred between different tasks in the same environment.

In order to disentangle the learning of skill and knowledge, we need to adapt the way in which we present training examples to the agent, as well as the architecture of the model. Knowing that all tasks can be represented as a combination of skill and knowledge, we take inspiration from recent work on disentangled Variational Auto Encoder (VAE) [107] to learn a joint latent representation across all the tasks to be performed. This latent representation is partitioned into

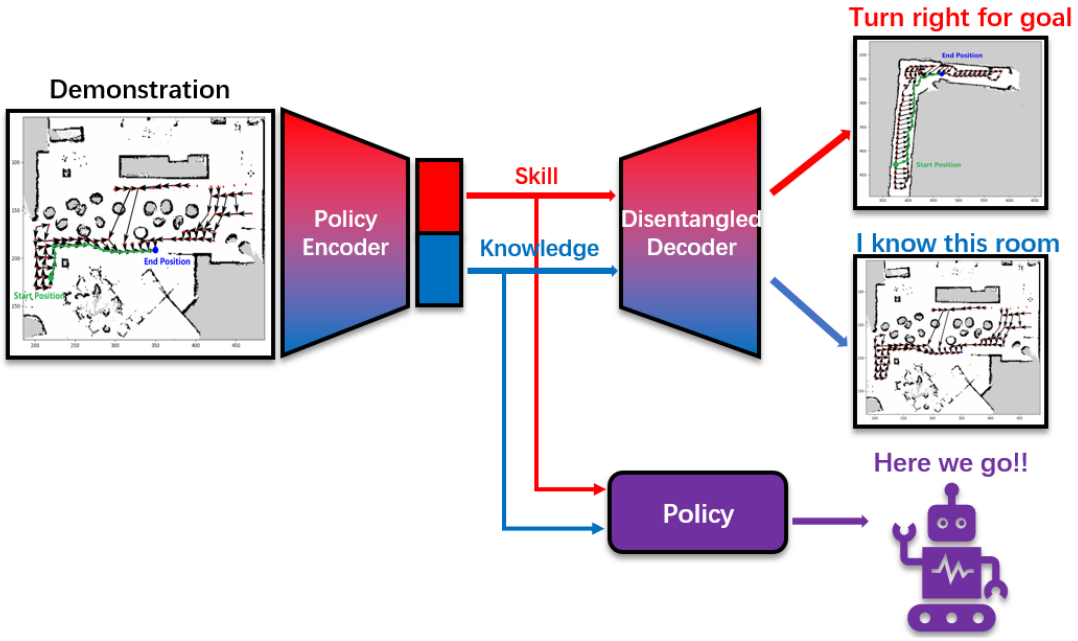


Figure 4.1: The policy encoder provides an embedding consisting of both skill and knowledge, coupled with the disentangled decoder to form a gated VAE architecture which partitions the embedded latent.

two subdomains dedicated to representing the skill and the knowledge of the task. These latent subdomains are jointly trained in a weakly supervised manner, in parallel to learning a policy from the latent observation space.

We show experimentally that we are able to successfully disentangle the skill and knowledge in multi-task learning, Furthermore, we show that this improves training efficiency and final performance. To summarise, the main contributions of this study are as follows:

- A self-supervised VAE-based architecture to learn a disentangled representation of robotic tasks
- A multi-task imitation Learning approach which shares training experiences across latent subdomains
- An approach to generate a more human-interpretable latent space for multi-task imitation learning, enabling decoding and visualization of the latent for better understanding.

A preliminary version of this work was published in IROS 2022 [115]. The rest of this

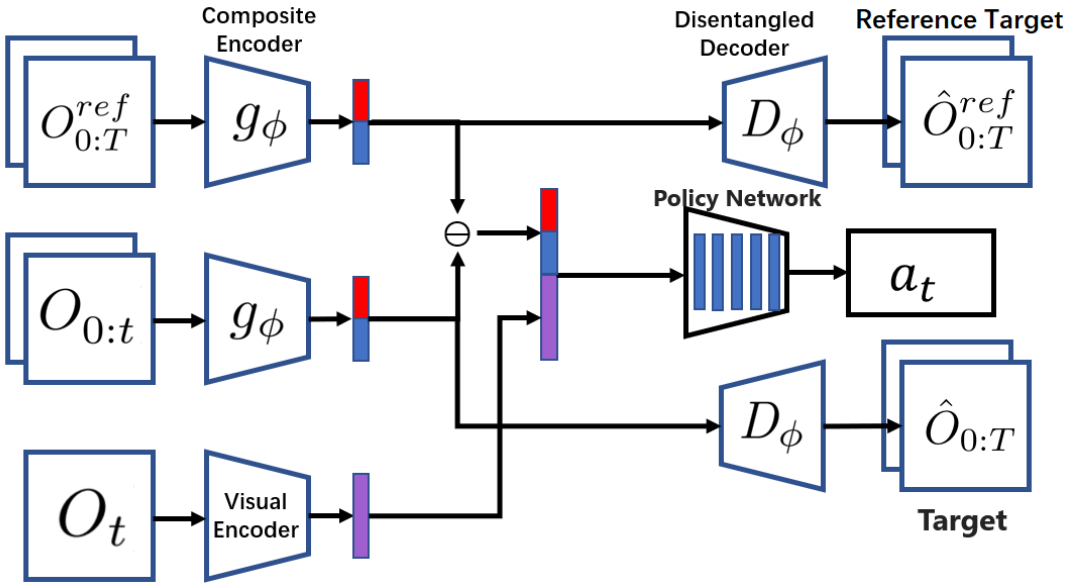


Figure 4.2: The network requires a current state O_t, a reference trajectory $O^{ref}_{0:T}$, and the current trajectory $O_{0:t}$. The output consists of both an action and two reconstructed image pairs for the reference input and the current input.

chapter is organised as follows: Section 4.2 introduces our methodology, and formalizes the disentangling technique and the training framework. Section 4.3 evaluation the results of our various experiments, and we conclude this work in Section 4.4.

4.2 Methodology

In this chapter, we introduce the Skill and Knowledge Independent Latent Learning (SKILL) architecture, as shown in figure 4.2. With this architecture and training framework, we attempt to disentangle the learned skill and knowledge within a latent embedding. The architecture consists of a pair of gated VAEs [107] that share weights, the latent spaces of these VAEs are partitioned into skill and knowledge subdomains. Each subdomain includes a masked latent similarity loss between pairs of examples within the batch. The VAE is given two sets of trajectories which consist of the start and end state as well as the current point of the agent. These are used to produce a CPV which plans from the current state to the end state. The gated VAEs are trained using pairs of experiences which share either the same environment, the same task list, or both, in order to disentangle their latent space. Similar to a human, learning how to drive a car by

driving to different places, and learning the city's layout by travelling around the city using different modes of transport.

The agent is required to perform tasks, which include modifying its environment to reach a target goal state. The agent may need to complete several subtasks in order to complete a task. The sequence of states visited during the completion of the task is referred to as the trajectory of the agent. The input does not specify any particular ordering of the subtasks within the trajectory. The set of subtasks is implicit, which gives the agent the freedom to determine the best set of subtasks to reach a particular goal state.

The embedding of the portion of the task remaining to complete (current-to-end) is calculated as the difference between the embedding of the overall task (start-to-end), and the embedding of the progress so far (start-to-current). This is combined with a visual embedding and passed to the policy network to determine the agent's next action.

In our experiments we perform imitation learning, taking the full set of states from timestep $t = 0$ to the final timestep $t = T$, let O be the observation of a state in a fully observable environment, then $O_0^{ref}...O_T^{ref}$ is an expert reference trajectory. The expert reference trajectories are extracted by a greedy search over the environment for the optimal solution. However, the proposed approach is equally applicable to unguided reinforcement learning

4.2.1 Compositional Task Embedding

In order to learn specifically the skill partition of the latent space, the agent will be required to complete multi-task learning without changing the training environment. This will ensure the environment has minimum impact on the learning process. Additionally, a latent which embeds both skill and knowledge of a single task is required. To acquire such latent, we will first define a compositional representation for any combination of sub-tasks in a multi-task learning environment. A compositional representation is an embedding which encodes structural relationships between the items in the space [73]. The multi-task environment will help provide the disentanglement by allowing tasks and environments to be mixed in different combinations. Consider a compositional task embedding $\vec{v}$ which encodes a set of tasks as the sum of the compositional embeddings for all subtasks. To avoid enforcing a particular ordering for the completion of these subtasks, our planning space is built with commutativity, i.e. A+B =

B+A. Given this definition, the embedding of all tasks that have yet to be accomplished can be calculated as $(\vec{v} - \vec{u})$ where $\vec{u}$ is the embedding vector for the tasks accomplished so far. As we focus on semi-supervised machine learning, we don't specify the exact end state for the agent. Instead, the policy $\pi(a_t|O_t, \vec{v} - \vec{u})$ produces the action a_t based on the current state O_t and the "to do" task embedding.

Next, we introduce the related losses for the model. Let function $g_\phi(O_{a:b})$ encode the observation pair at time a and b into a latent task embedding using parameters ϕ. The latent vectors $\vec{v}$ and $\vec{u}$ will be the target of the disentangling process as each of them should contain both the learned skill which solves the task as well as the specific knowledge of the corresponding environment. To help further the learning of the task embedding, the function g is a probabilistic encoder which predicts means and variances for each latent parameter. This is coupled with a decoder d_θ to form a VAE, such that $O \approx d_\theta(\vec{u})$ where $\vec{u} \sim g_\phi(O)$. We define the reconstruction error against target $\hat{O}$ as $l_{rec}(O, \hat{O}) = |d_\theta(g_\phi(O) - \hat{O})|$ where the intermediate sampling step is omitted for brevity. The full reconstruction loss is obtained by applying this to both the reference trajectory $(O^{ref}_{0:T})$ and current trajectory $(O_{0:t})$ inputs:

$$L_\delta(O^{ref}_{0:T}, O_{0:t}, \hat{O}^{ref}, \hat{O}) = l_\delta\left(O^{ref}_{0:T}, \hat{O}^{ref}\right) + l_\delta\left(O_{0:t}, \hat{O}\right) \tag{4.1}$$

To reduce the impact of empty space, we also mask the reconstruction loss to only include non-zero pixels.

During the forward pass, as both skill and knowledge are required to solve the task, the entire latent space is used by the policy network to select an action. Therefore, the policy function is:

$$\pi\left(a_t|O_t, g_\phi\left(O^{ref}_{0:T}\right) - g_\phi\left(O_{0:t}\right)\right) \tag{4.2}$$

Hence the policy loss L_a is given by the loss function:

$$L_a(O_t, \phi) = -log\left(\pi\left(\hat{a}_t|O_t, g_\phi\left(O^{ref}_{0:T}\right) - g_\phi(O_{0:t})\right)\right) \tag{4.3}$$

where $\hat{a}_t$ is the reference action.

The policy action is essentially being predicted based on the difference between the embedding of reference trajectory $O^{ref}_{0:T}$ and the agent's current trajectory $O^t_{0:t}$. The task embedding is given by function g_ϕ.

Additionally, there are two regularization losses using the triplet margin loss l_m from [91]. The first L_H enforces the compositionality of the latent space by ensuring that the sum of the embeddings for partial completion ($u_{0:t}$) and the embedded to-do vector ($u_{t:T}$) are equal to the embedding for the entire task ($u_{0:T}$).

$$L_H(O_0, O_t, O_T) = l_m(g_\phi(O_{0:t}) + g_\phi(O^{ref}_{t:T}) - g_\phi(O^{ref}_{0:T})) \tag{4.4}$$

where l_m is a truncated L1 loss with a margin equal to 1. The second regularization loss L_P tries to ensure that similarity in the latent space corresponds to semantically similar tasks. To this end, we ensure that the embedding of our agent's trajectory is similar to that of the embedding of the expert's reference trajectory

$$L_P(O_0, O_t, O_T) = l_m(g_\phi(O_{0:T}) - g_\phi(O^{ref}_{0:T})) \tag{4.5}$$

The sum of these two loss functions is used to regularize the model:

$$L_R = L_H + L_P \tag{4.6}$$

The latent representation used by the agent comprises both the ability to solve the current task, which is the skill, and the information about the current environment, which is the knowledge. However, these two types of latent information are currently entangled. This loss function does not provide any capability for disentangling the latent representations.

4.2.2 Gated Variational Auto Encoders

To disentangle the task vectors ($\vec{u}$) into skill and knowledge sub-domains($\vec{u} = [\vec{u}^s, \vec{u}^k]$), we utilize the gated VAE [107] approach with the CPV encoders as part of the VAE.

Gated-VAE is a weakly-supervised approach for training VAEs, through the gating of the backpropagation process. The gradients flowing through a subdomain of the latent space are blocked while the other subdomains update normally. Which subdomains are gated, is determined based on the shared properties of the input and target image pairs. In this work, the gating is determined by training dataset $X \in (\mathcal{S}, \mathcal{K}, \mathcal{N})$. Where $\mathcal{S}$ is the skill training set, $\mathcal{K}$ is the knowledge training set, and $\mathcal{N}$ is the normal training set such that $\mathcal{N} = \mathcal{S} \cap \mathcal{K}$.

In a standard VAE, we have a dataset x, parametrised by ground truth generative factor z, the encoder and decoder are parametrised by ϕ and θ receptively, and the goal generative process can be described as:

$$\underset{\phi,\theta}{max}\ \mathbb{E}_{q_{\phi,\theta}(z|x)}[log\ p_\theta(x|z)] \tag{4.7}$$

Then the objective function can be re-written as:

$$L(\phi,\theta;x,z) = \mathbb{E}_{q_{\phi,\theta}(z|x)}[log\ p_\theta(x|z) - D_{KL}(q_{\phi,\theta}(z|x)||p(z))] \tag{4.8}$$

Where D_{KL} is the non-negative Kullback-Leibler (KL) divergence for regularizing the approximation of $q_{\phi,\theta}(z|x)$.

In this work, we use β-VAE as the backbone VAE, which is a modified variation of the VAE framework with an additional hyperparameter β in the objective function to produce the reconstruction loss function:

$$L_\delta = L(\phi,\theta;x,z) = \mathbb{E}_{q_{\phi,\theta}(z|x)}[logp_\theta(x|z) - \beta\ D_{KL}(q_{\phi,\theta}(z|x)||p(z))] \tag{4.9}$$

The term $log\ p_\theta(x|z)$ encourages the reconstruction accuracy, while the KL divergence term with $\beta(usually\beta > 1)$ gives more weight to encourage disentanglement. [18]

A gated VAE utilizes the intuition that the input and target images need not be identical, but can instead be paired according to shared factors. By incorporating this pairing into the training process, we introduce additional supervision into the VAE model. The network should be able to learn and recognise the shared factor within the pairing and be further encouraged to disentangle the shared factors. During forward propagation, all partitions of the latent space are used by concatenating all the partitions and computing the KL divergence loss over the entire latent. The backpropagation process can then be either gated or allowed through depending on the specific latent partition corresponding to the related shared factors.

The training data will be split into three groups where the input/target pairs are different ($x_{input} \neq x_{target}$) but share a common factor h ($\vec{s}_{h,x_{input}} = \vec{s}_{h,x_{target}}\ \forall h$). This will allow the partition to commence and disentangle the latent space $\vec{u}$ into a set of factors $[\vec{s}] \subset \vec{u}$.

In the context of multi-task learning, we define the shared factors being the current task the agent is performing, and the environment the agent is in. With this definition, when an agent is

performing the same task between the input and target, the skill for solving this task is a shared factor. Respectively, when the agent is performing a different task in the same environment, the knowledge of the environment is the shared factor.

4.2.3 Disentangling Skill and Knowledge Subdomains

To disentangle the learned latent, we can generate the training examples which pair the input/target images according to the shared skill and knowledge factors. Specifically, if two training examples (O and $\hat{O}$) both comprise the same sequence of subtasks but within a different environment, these examples are grouped by skill and added to the skill training set $\mathcal{S} = \mathcal{S} \cup (O, \hat{O})$. Similarly, if the training examples comprise different sequences of subtasks, but within the same environment, they are grouped by knowledge and added to the knowledge training set $\mathcal{K} = \mathcal{K} \cup (O, \hat{O})$. In this work we enforce a hard gating by partitioning the latent space into two non-overlapping regions, the ratio of the sizes of these two latent subdomains can be changed based on the task. In all our experiments we kept them equal, each representing either skill or knowledge.

To disentangle the skill from knowledge, we adapt the reconstruction loss from equation 1. The input and target pair for both terms are drawn from either the skill or knowledge training set such that $(O, \hat{O}) \in (\mathcal{S} \cup \mathcal{K})$. We additionally adapt which partition of the latent space is updated via backpropagation based on this. For every training example, we select the training mode $\mathcal{G} \in (\mathcal{S}, \mathcal{K}, \mathcal{N})$ and select an input/target pair from within the corresponding dataset. Then a learned full latent $\vec{u}$ can be split into the skill $\vec{u}_s$ and knowledge $\vec{u}_t$ latent. With gating mode $\mathcal{G}$, which turns off the gradient flow for $\vec{u}_s$ or $\vec{u}_k$ during back-propagation.

This means that for each training pair, gradient flow and parameter updates only occur for the subdivision of the latent space which is shared by the source and the target. For example, in one skill training iteration, the input O is paired with target image $\hat{O}$, with the training mode being skill ($\mathcal{G} = \mathcal{S}$, and hence $(O, \hat{O}) \in \mathcal{S}$). The composite plan vector $\vec{u}$, which is the concatenation of the skill sub-domain partition $\vec{u}_s$ and the knowledge sub-domain partition $\vec{u}_k$, will only accept the gradients from $\vec{u}_s$ in the back pass. On the other hand, in knowledge training iteration ($\mathcal{G} = \mathcal{K}$, $(O, \hat{O}) \in \mathcal{K}$), the gradients from the skill sub-domain $\vec{u}_s$ will be masked.

More formally, we define $[\![]\!]$ as an operator which masks gradients during the backpropagation.

We then define the gated latent space as:

$$\vec{u} = \begin{cases} [\vec{u}_s, [\![\vec{u}_k]\!]] & if(O, \hat{O}) \in \mathcal{S} \\ [[\![\vec{u}_s]\!], \vec{u}_k] & if(O, \hat{O}) \in \mathcal{K} \\ [\vec{u}_s, \vec{u}_k] & if(O, \hat{O}) \in \mathcal{S} \cap \mathcal{K} \end{cases} \tag{4.10}$$

With this gated training framework, we are able to cluster examples based on shared subdomains. It is worth noting that within our framework the grouping and subsequent selection of skill or knowledge targets are done for both the current branch $(O, \hat{O})$ and the reference branch $(O^{ref}, \hat{O}^{ref})$.

Additionally, we introduce a dynamic loss $L_{\mathcal{G}}$. At the end of each training iteration, the total loss consists of the action loss from the policy network L_a and the reconstruction loss L_δ. The action loss of the agent is more closely related to the skill which solves the task and thus should be weighted more in skill training mode. The reconstruction loss in skill training mode will be large in value (as the entire environment differs between input and target). However, this loss is weakly correlated to the learning of skills. A similar situation also exists in the opposite scenario. In knowledge training mode, the agent will perform different tasks, which will cause a surge in action loss and potentially have an adverse effect on the knowledge latent space. To counter this effect, we introduce regularization constants α and β which balance the numerical value of action loss and reconstruction loss. Additionally, while α, β are regularization constants, it is possible to improve the disentanglement performance by changing the value of α and β according to the training mode $\mathcal{G}$. This dynamical loss can be expressed as:

$$L_{\mathcal{G}} = \begin{cases} \epsilon\alpha L_a + \beta L_\delta & if(O, \hat{O}) \in \mathcal{S} \\ \alpha L_a + \epsilon\beta L_\delta & if(O, \hat{O}) \in \mathcal{K} \\ \alpha L_a + \beta L_\delta & if(O, \hat{O}) \in S \cap K \end{cases} \tag{4.11}$$

where ϵ is a small value constant, down-weighting the less relevant loss for each scenario.

To summarize, the loss function L of the framework comprises both reconstruction loss and policy loss with the dynamic loss weighting $L_{\mathcal{G}}$. This is summed with the regularization loss

L_R with a value adjustment constant ω to be at the same numerical level as a dynamic loss:

$$L = L_G + \omega L_R \tag{4.12}$$

4.3 Evaluation

We evaluate the SKILL framework to show how the proposed disentanglement of skill and knowledge impacts both the agent's success rate and efficiency. We perform a range of qualitative experiments, exploring and confirming the level of disentanglement learned by our system. Following this, we explore the importance of different elements of our system via an ablation study. We also evaluate this across two different environments and compare it against the current state-of-the-art technique in each. Finally, we demonstrate our technique with a real robot performing navigation tasks.

Craftworld Environment The first environment used in our experiments is a Minecraft-inspired 2D crafting world [32]. The world has a discrete state and action. The agent is able to move, pick up or drop off certain items present on the map, as well as perform actions on those items. With this environment, we can define tasks such as chop tree, break rock, make bread, build a house, etc. and combine them into sequences such as [make bread, eat bread, chop a tree, build house]. This provides a good selection of unique tasks and sequences to generate training data. For example, the agent will be able to cut down trees if it has previously picked up an axe, and will be able to use an oven to bake bread if it has previously harvested wheat. As detailed in the methodology section, the objectives of the agent are specified implicitly by providing two trajectories consisting of observations of both the current trajectory and the reference trajectory. The advantage of this approach is that no explicit ordering of subtasks is specified, and the agent is free to execute tasks in the most appropriate manner. Our framework is trained with randomly generated starting environments and random combinations of tasks to complete. The complexity of the problem increases as more tasks are required to reach the target end state. The previous state-of-the-art approach in this environment [33] used the same input observations and expert reference trajectories.

The model is given three sets of data as shown in figure 4.3. Firstly, an original episode with an environment and a sequence of tasks. Secondly, an episode with the same environment but

Figure 4.3: The different inputs for different training modes. In skill mode, the environment differs from the original but the agent is expected to perform the same task. In knowledge mode, the environment is the same but the agent is expected to perform a different task.

different tasks for knowledge training. Thirdly, an episode with the same tasks and a different environment for skill training. In figure 4.3, the *original* episode requires the agent to pick up a hammer and break a rock. In the *knowledge training* episode, the environment is the same, but the task is to pick up wheat and make bread. In the *skill training* episode, the environment is different from the original episode, but the task is once again to use a hammer to break the rock. Depending on the training mode, while the agent will always be asked to perform the task given in the original episode, the demonstration given will change.

Learned Navigation The second environment simulates a 2D navigation scenario. The maps are created from gmapping [36] outputs in real-world locations to simulate real-world navigation as shown in figure 4.4. The goal in this environment is to reach a random target location on the map. The agent is given a full state observation as well as a demonstration during training. In both environments, we focus on two evaluation metrics: the task success rate measures how many episodes end in the goal is successfully reached. The average episode length measures how quickly the agent was able to achieve its goal.

4.3.1 Implementation

In both environments, the observation is provided as a pair of images. The encoder g shared by the reference trajectories $O^{ref}_{0:T}$ and the current trajectories $O_{0:t}$ is a 4-layer CNN encoder with

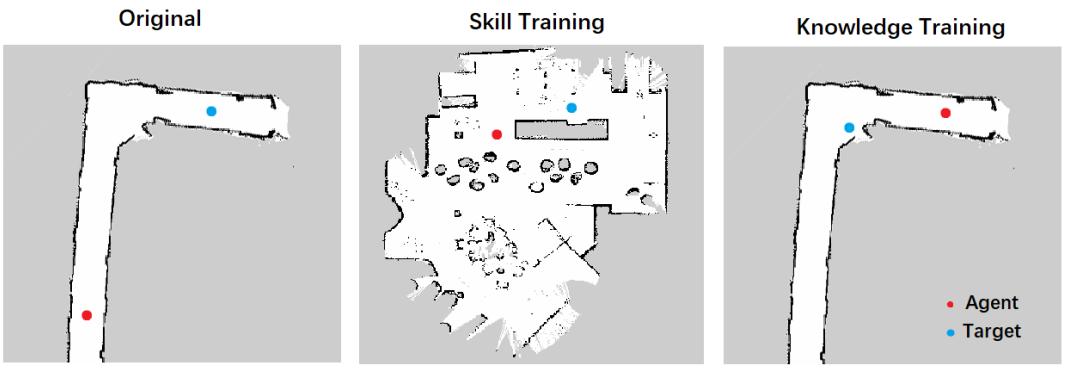


Figure 4.4: The navigation environment mimics real maps produced by the gmapping [36] algorithm.

shared weights. The current state input (O_t) is processed by a 4-layer CNN with a final fully connected layer. The last layer of the encoder branches out according to the ratio between the skill and knowledge latent. The latent produced by both branches are concatenated before the subtraction of $g_\phi(O^{ref}_{0:T}) - g_\phi(O_{0:t})$. The resulting latent embedding of the task left to complete is fed into a 5-layer policy network to produce the action during each time step, while each disentangled latent is fed into a 4-layer decoder for reconstruction.

With the different training modes and latent gating, we are able to control and disentangle the skill and knowledge learned during training. Our approach will increase the efficiency of multi-task learning and transfer learning since specific knowledge on the current task or environment can be learned independently. It is worth noting that to achieve this, we used hard gating for the disentanglement of the latent. This assumes that the skill and knowledge used to solve any task can be completely separated. It is arguable to what degree this is strictly true as to perform any task, both skill and knowledge are required, and they are always entangled on a certain level.

4.3.2 Exploring Disentanglement

In our first set of experiments, we seek to confirm whether our proposed approach results in a latent space where skill and knowledge are disentangled. Unfortunately, measuring disentanglement is extremely challenging. There are many proposed approaches in the literature but most require the ground-truth factors to be known. Instead, to quantify our disentanglement of

skill and knowledge, we first take our trained model and freeze the network weights. Next, we record the latent embeddings produced by our network for all samples in the dataset. Finally, we attempt to train a simple network that estimates the task id from only one of the latent partitions ($\vec{u}^s$ or $\vec{u}^k$).

When testing on a held-out set of 500 unseen latent embeddings, the network trained on the skill partition is able to achieve 99.2% accuracy in recovering 6 different task labels. However, for the network trained on the knowledge latent, the recovery accuracy is only 12.7%. This shows that all the information pertaining to the tasks to be completed has been effectively disentangled, and concentrated into the latent skill subdomain. This indicates the skill latent has a significant difference in the distribution to the knowledge latent. To ensure the recovery network didn't simply pick up characteristics from the spatial difference, we used more than one model when producing the latent, the results remains similar.

We could not perform a similar test for the knowledge subdomain because we do not have a fixed number of environmental layouts to recognise. Instead, we trained two image decoder networks which attempt to reconstruct the environment using only the skill or knowledge latent partitions respectively. The decoder networks use the latent partitions as input to generate the corresponding observations, both are trained until they reach their peak accuracy. Examples of reconstructed images for previously unseen latent embeddings are shown in figure 4.5.

It is apparent that the reconstruction results using the previously unseen knowledge latent are much better than the results from the skill latent, We note that the skill latent alone is unable to produce any meaningful image. Meanwhile, the knowledge-only reconstructions appear to focus on representing the most salient parts of the environment. The reconstruction from the full latent is able to reconstruct the environment fully. This shows that some of the less salient final details may be jointly encoded across both latent subdomains. This mirrors findings in neuroscience which indicate that in biological learning, declarative and procedural knowledge can be disentangled to a great extent but never completely.

Numerically, the average reconstruction loss across the validation dataset for the knowledge latent is around 300 times lower than the reconstruction loss from the skill latent. We find similar results in the second environment, where the average reconstruction loss across the knowledge latent dataset is around 250 times lower than with the skill latent.

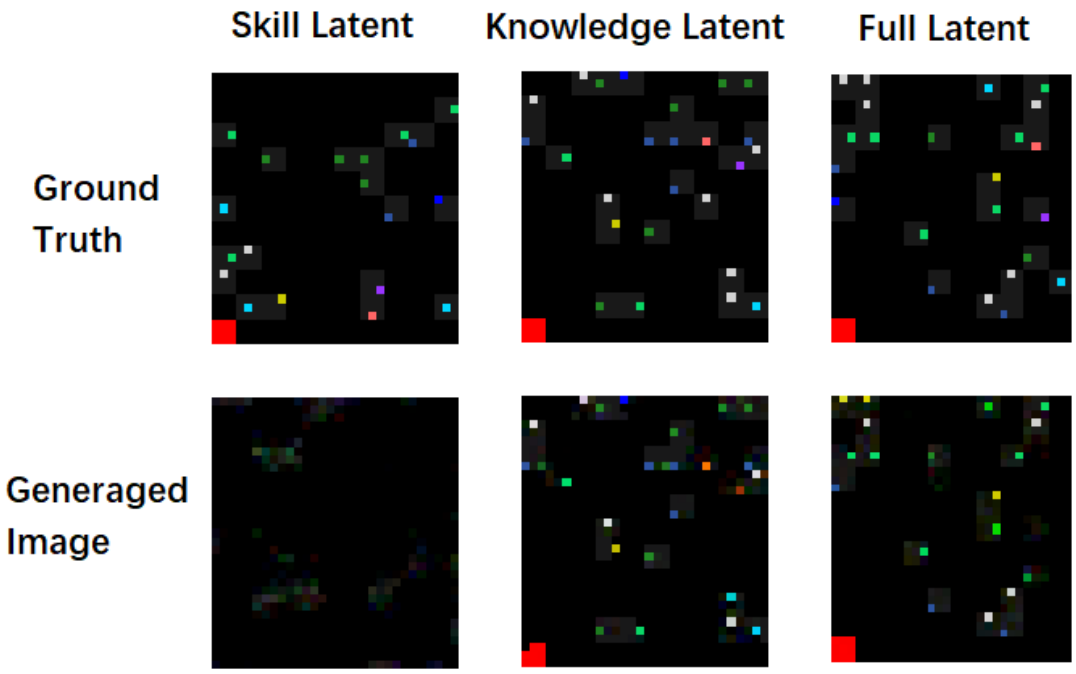


Figure 4.5: The reconstructed images using the knowledge or full latent are similar to the target image, while the reconstructed images produced by skill latent do not resemble any characteristics of the target image.

We consider these results a sign of successful disentanglement of skill and knowledge in the latent embedding. However, further experiment shows there is still entanglement between skill and knowledge. We are not able to fully disentangle with the current hard gating model.

For our final experiment to explore the disentanglement, we combine the latent skill partition from one example and the latent knowledge partition from a different example. We then push the resultant full embedding through the decoder and examine the resulting image. This experiment is performed to explore our assumptions regarding the hard latent partition. A typical result of this experiment is shown in figure 4.5. It is worth noting that this type of task recombination is a form of zero-shot learning, as the specific combination of task and environment will not have been seen during training.

Although generally correct, the resulting reconstruction often demonstrates a combination of characteristics from both donors latent, rather than a perfect reconstruction of the environment from the knowledge partition. As shown in figure 4.6, the resulting state reconstruction contains features from both donor states, circled in blue and red. A complete disentanglement in the

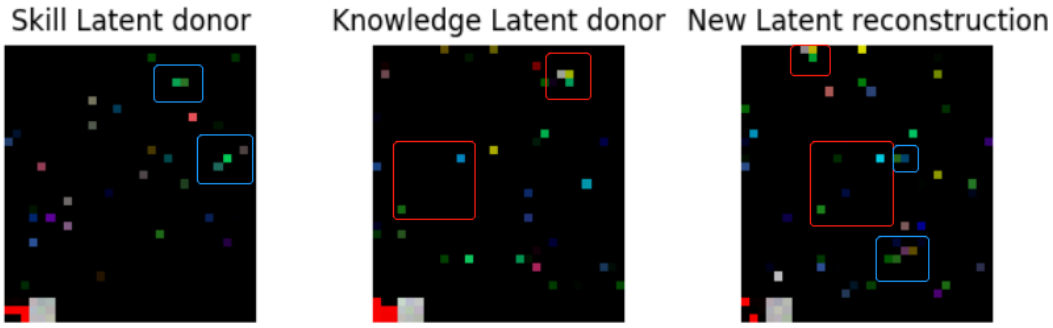


Figure 4.6: The resulting reconstruction shows shifted characteristics from both donors. However, the overall similarity to either donor's reconstruction is generally low.

latent subdomain should have resulted in a reconstructed state exactly the same as the knowledge latent donor. However, our results showed a rather low similarity to the knowledge latent donor when mix-matching, while the same network would reconstruct the state with a full latent perfectly. This indicates our hard-gating approach to disentangling the skill and knowledge cannot fully disentangle all information within the latent. Certain environmental information is still preserved in the skill latent.

4.3.3 Ablation Study

Model	Imitation accuracy	Success	Ep. length
CPV-FULL[33]	66.42%	65%	69.31
SKILL-no O_t	64.18%	65%	26.95
SKILL	70.61%	84%	19.77
SKILL+FS	**70.89%**	89%	19.52
SKILL+FS+DL	70.62%	**94%**	**17.88**

Table 4.1: Ablation study. FS indicates fixed sampling. DL indicates dynamic loss weighting.

Now that we have conclusively demonstrated the successful disentanglement of our learned

Model	Success	Ep. Length
SKILL+FS	90%	14.82
SKILL+FS+DL	96%	13.08
SKILL+FS+KP	**98%**	13.31
SKILL+FS+SP	84%	**11.47**

Table 4.2: Ablation study for Env.2. KP indicates higher knowledge partition, SP indicates higher skill partition.

embedding space, we next perform an ablation analysis of our system. To this end, we explore the contributions of 3 parts of our model. For this experiment we additionally report the imitation accuracy (percentage of actions that agree with the expert) for comparison against [33]. As shown in table 4.1, we first removed the current state observation branch (no O_t). With only the gated VAE structure, our model performs similarly to the SOTA model (CPV-FULL[33]). Introducing the current state observation branch improves performance significantly by giving the agent a more direct observation of its current state. In the next experiment, we remove the random sampling from the latent distribution, and instead simply take the mean latent embedding. We refer to this as Fixed Sampling (FS). This offers a small improvement in all metrics. Finally, we introduce the dynamic loss (DL) weighting scheme proposed in equation 4.11. This approach provides further improvement in task performance and completion speed. The imitation accuracy maximized at around 70% while the task performance continues to increase. Adjusting the proportions of the loss functions according to the training mode improves training stability. However, this also reduces the training speed, as the learning happens less aggressively. It is interesting to note that although imitation accuracy goes down slightly for the DL model, task success and completion speed go up. This indicates that our agent may now be outperforming the expert.

In the second environment, we study the effect of different partition ratios shown in table 4.2. We used FS and FS+DL as references, with the latent space split evenly between skill and knowledge. When we allocate more of the latent space to the knowledge subdomain (KP), the

result surpasses FS+DL model in both success rate and speed. When a higher partition is given to skill subdomain (SP), while the task success rate dropped by 15%, the completion speed increased significantly. This indicates an interesting trade-off between environmental knowledge for successful navigation and skill for efficiency.

4.3.4 Comparison vs State-Of-The-Art

MODEL	4 SKILLS		8 SKILLS		16 SKILLS	
	success	ep. Length	success	ep. Length	success	ep. Length
CPV-NAÏVE[33]	52.5	82.3	29.4	157.9	17.5	328.9
CPV-FULL[33]	71.8	83.3	37.3	142.8	22	295.8
SKILL	**61.3**	**63.3**	**37.5**	**132.7**	**20**	**277.8**

Table 4.3: Comparing against SOTA when learning a different number of tasks.

MODEL	1,1		2,2		4,4	
	success	ep. Length	success	ep. Length	success	ep. Length
CPV-NAÏVE[33]	57.5	**36**	0	–	0	–
CPV-FULL[33]	73	69.3	**58**	270.2	20	379.8
SKILL	**80**	53.3	55	**103.1**	**26.3**	**198.1**

Table 4.4: Compared against SOTA when learning a different number of tasks within the sequence.

After determining the optimal approach, we will now compare our model more thoroughly against the previous SOTA model (CPV-FULL[33]) in both environments. As we used CPV as the backbone network, it is the natural choice for examining the effect of the disentanglement. For craftworld [32], we follow the evaluation protocol [33]. Both our model and the SOTA model are trained on 50,000 samples from sequences with 1-3 different tasks, and we evaluate each model against sequences with 4,8, and 16 different tasks, with results shown in table 4.3. Our model outperforms the SOTA model in both task success rate as well as performance speed in most cases. In particular, our technique leads to a 30% relative increase in the success rate of the most challenging experiment, and a 50% reduction in episode length.

We also evaluated the model's generalization capability with sequence of tasks. Experiments with "1,1" being a single task, and "2,2" being a sequence of 2 tasks from 2 reference trajectories summed together. As shown in table 4.4, the disentangled model has a significant advantage in performance over the CPV-NAÏVE[33] model. While the success percentage is slightly lower than the CPV-FULL model in the "2,2" scenario, the SKILL model still has a lower step count average per episode and better performance overall in the "4,4" scenario. This indicates that our model has a better generalization capability when dealing with trajectories with more tasks, as well as when dealing with a sequence of tasks.

In the navigation environment (figure 4.4), we compare with our previous work [114] trained under the same conditions. The MERLIN model is adjusted to accept the state image as input. The number of environments under training is increased to 6 different navigation environments. The agent is required to navigate between two random spots in each environment, the average success rate and step count are used as a performance benchmark. The MERLIN has a success rate of 94.6% and an average step count of 19.96. The SKILL model is able to achieve a 98.0% success rate with an average step count of 14.58. Hence the average efficiency of our agent is also 20%-30% faster than then the previous SOTA.

With the disentanglement of skill and knowledge, we are able to better share useful experiences across different navigation tasks. Interestingly, we note that when attempting to generalize to a longer sequence of tasks, the performance of the SKILL model and the CPV model both show a significant drop when handling longer sequences of tasks. Though both SKILL models maintain a better performance than the CPV model, the task success rate almost halved when moving from 2 task sequences to 4 task sequences. This drop shows the current SKILL model has difficulty in long-horizon planning, as well as a limited capability in sequence generalization.

4.3.5 Real Life Demonstration

Lastly, we demonstrate our model with a live turtlebot3[2] as the robotic platform. The turtlebot first creates a map of the area using gmapping[36], which is then processed into an observation format recognizable by the agent. The agent we used for this experiment is trained in simulated navigation environments. The target location is marked on the observation along with the robot's current location. The current location is obtained with lidar-based localization. At each timestep,

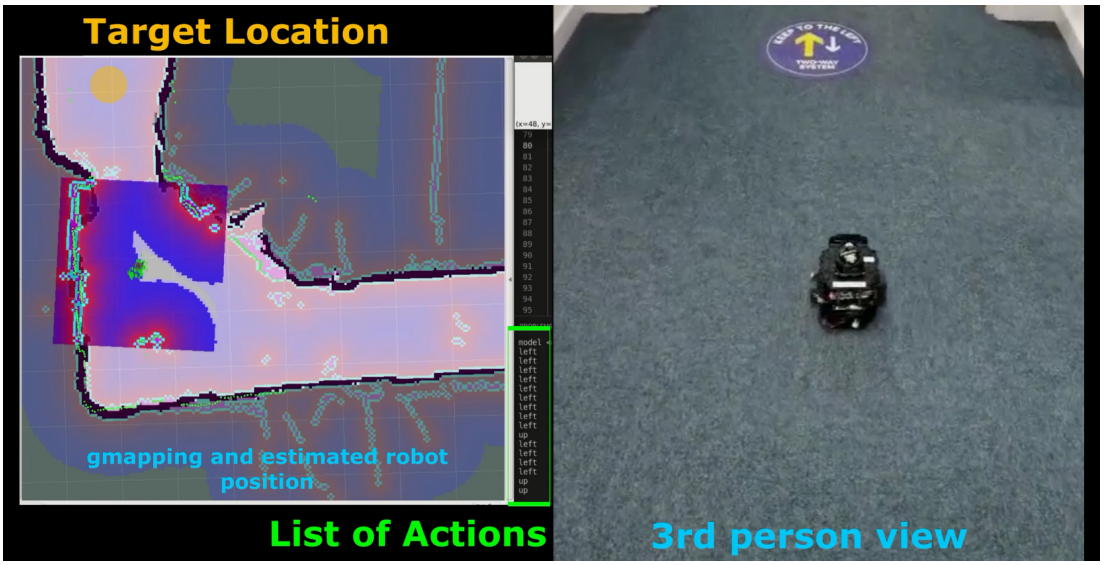


Figure 4.7: Live demonstration of SKILL-IL agent, the Turtlebot3 follows instructions predicted by the agent to reach a goal location.

the agent will produce a command for the robot to go in one of four directions for a set distance. The navigation method used is dead reckoning as we expect the drift to be compensated by the agent when making the decision. This process repeats until the robot reaches the target location.

Without any fine-tuning, the robot is able to compensate for odometry inaccuracies and drift by performing recovery moves during the navigation, even though it was not exposed to this drift during the simulated training. Our robot is able to successfully complete the task in multiple locations, a screen capture of the demonstration video is shown in figure 4.7.

4.4 Conclusion

In this work, we approached the problem of multi-task learning from a new perspective. Taking inspiration from neurobiology and pedagogy on memory acquisition, we hypothesized the latent space in a policy neural network could be disentangled into subdomains. Each partition is responsible for either the skill or the knowledge of the task and should be transferable to different combinations of future experiences. We successfully demonstrated this disentanglement in imitation learning, using a gated VAE architecture. With our method, we are able to outperform the SOTA model, in two different environments, both in terms of success rate and speed.

During our experiments, we discover the entanglement between skill and knowledge can not be completely separated, this finding echoes the studies in pedagogy and language acquisition. Even for humans, certain procedural memory is entangled with declarative memory.

Nevertheless, being able to disentangle the skill and knowledge in a task is a fundamental step toward combinational generalization. A better model to partition the skill and knowledge latent or to explain the entangled information will benefit our understanding of imitation learning in general. Human interpretable solutions to complex tasks are also an interesting direction as it's been a popular choice in multi-task learning.

This chapter's work in disentanglement provided the possibility for modularity in learned latent. This led to drastically improved performance compared to previous state-of-the-art. However, in this chapter's experiments, we find a significant performance drop when dealing with a long sequence of tasks. This presents a limitation in the generalization capability as well as challenge in long horizon planning. Additionally, the experiments are only performed in a 2D toy environment. There is no clear indication that this approach will be able to disentangle skill and knowledge in more complex and realistic tasks and environments. In the next chapter, we will improve upon this and further explore multi-task learning in terms of long, complex task solving and task sequence generalization.

Chapter 5

Compositional Adaptive Subgoal Estimation for One-Shot Task Generalization

5.1 Problem Definition

In the previous chapter, we explored the disentanglement of skill and knowledge in the learned latent representation. While we had successfully disentangled most of the skill and knowledge, a certain portion of the latent space remains entangled and cannot be separated with a discrete method. While experimenting with disentanglement, our agent is able to outperform the SOTA in both task success rate and generalization. However, when we test the agent's ability to generalize to longer task sequences, the agent performs poorly. This is likely caused by the implicit ordering of the tasks, as the agent focuses on individual tasks and largely ignored the reasoning between each task. For example, in order to *build a house*, the agent needs to gather lumber, which is the result of another task: use an axe to *chop a tree*. If the agent attempts to build the house before chopping the tree, then there's no lumber available for the agent and the task sequence will fail. Long-horizon complex tasks formed from a sequence of multiple subtasks have been a major challenge in reinforcement learning and imitation learning. In this chapter, we will explore the problem of solving long and complex tasks in sequence.

5.2 Introduction

As researchers seek to introduce robotic technology into various aspects of our society, robots must be able to perform increasingly complex tasks with enhanced automation and generality. These new tasks are often complex, with multiple implicit subgoals that vary depending on the environment. As such, it is common for only the target end goal to be specified explicitly. For example, if we ask the robot to bring us a cup of coffee, the robot will need to know where we are, as well as where the kitchen is, the tools, and the procedure for making coffee. The effort of learning such a complex composite task is enormous. More problematic is the fact that even if we provide explicit subgoal guidance: i.e. where our kitchen is, where the coffee machine is and how to use our coffee machine, this knowledge won't transfer to robots in other houses. Even for the individual robot, the solution may be brittle, as simply moving the location of the coffee cups may cause the task to fail.

The biggest learning challenge for solving complex tasks is the complexity itself. Any complex task would almost always require a large number of steps to successfully complete an episode. This is particularly true for tasks with terminal-only sparse rewards. The longer the average trajectory is, the broader we can expect an unbounded state space to become, and the lower our sample efficiency will be. In an Imitation Learning setting, the use of expert trajectories helps alleviate the "vanishing reward" problem by providing feedback at each step of the trajectory. However, the exploration and data efficiency problems remain.

The second challenge we seek to address is generalization. In an Imitation Learning setting, the data efficiency challenge mentioned above will often manifest as a relatively restricted set of expert trajectories. As such learning to perform a complex task often involves repetitively training on a small set of sample tasks. This can easily lead to over-fitting on the training task set or the specific training examples of the tasks. A common approach to mitigate this is to design the model hierarchically as shown in figure 5.1. In this case, each stage of the model is intended to specialize in solving a certain class of problems. This can simplify generalization within a subtask, but also exacerbates problems with data sparsity, as each sub-model will only be exposed to a small portion of the training data.

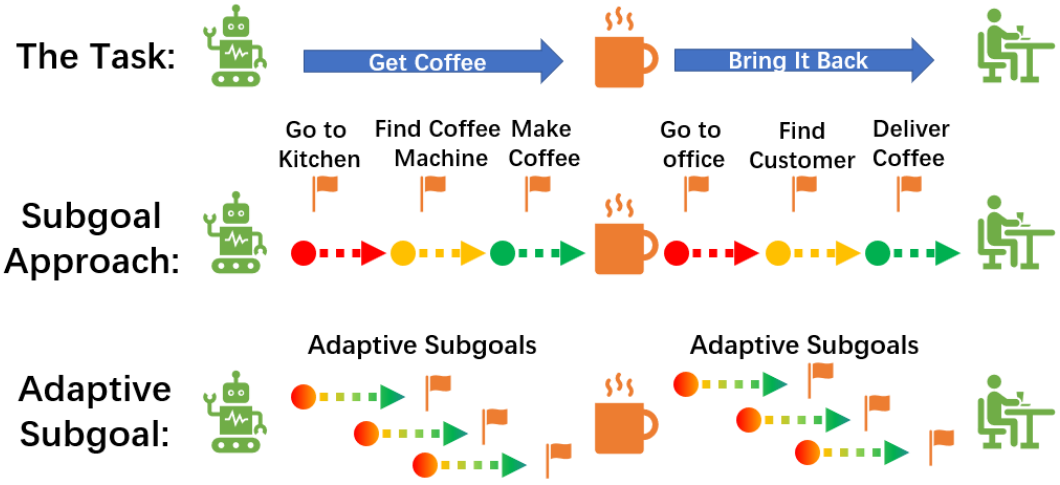


Figure 5.1: In the make coffee and bring it back task, the traditional subgoal approach segment the complex task into smaller, manageable subtasks. Our approach generates novel compositional subgoals at each step which gradually guides the agent towards the goal.

5.2.1 Exploring Task Compositionality

In order to solve these challenges, we hypothesise that we may be able to exploit the compositionality of the task embedding from the previous chapter. In the latent space where these representations reside, vector arithmetic can be applied to these task embeddings. As such it may be possible to generalise to previously unseen sequences of tasks, by composing the encoded subtasks which have been seen in other contexts.

In this chapter, we explore the potential of this compositional generalization. We first present preliminary experiments on the system from the previous chapter to assess the effectiveness of the learned compositionality. To this end, the skill subdomain of the latent representation ($\vec{u}_s$) should ensure that different embeddings of the same skill should be clustered closer than tasks representing other skills. On the other hand, the knowledge subdomain of the latent representation ($\vec{u}_k$) should cluster tasks according to their episodes.

As shown in figure 5.2, we explore the success of this compositional embedding by feeding a trained network with 50 episodes' states, where each episode contains a sequence of 2 to 8 tasks. This resulted in over 1000 compositional latent vectors and predicted actions. After visualizing

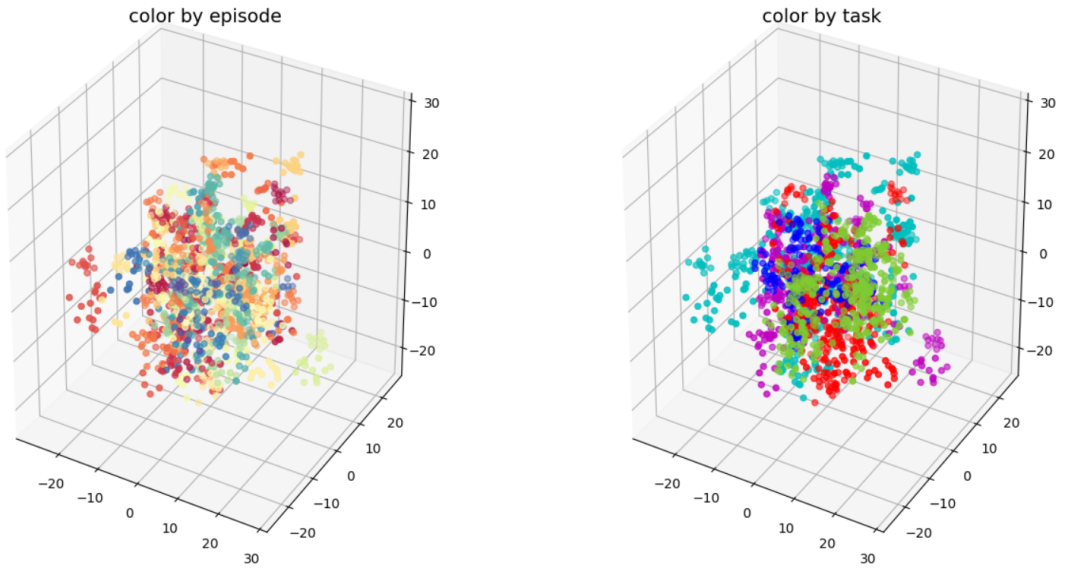


Figure 5.2: We encode 50 Episodes' states (over 1000 states) into compositional latent vectors with a trained network, and the result is projected into this 3D graph using PCA. The pointcloud is coloured by which environment the input state is from on the left side, and by which task the agent is performing on the right.

these data with PCA projection, we can clearly observe (on the right) some clustering of the representations based on the task. There is still some entanglement between the clusters, but this is likely due to the information loss of the PCA projection. The pointcloud coloured by episode (left) also exhibits clustering. However, this is less obvious as the number of episodes is far higher than the number of tasks, while the number of examples of each is comparatively lower. This is a clear indication similarity in tasks is being successfully encoded as similarity in the latent space. This is encouraging and implies that more complex combinatorial generalization may be possible.

In order to explore this possibility further, we investigate how the encoded plans are grouped according to task or action. For simplicity of visualization, we reduce the number of tasks within the data to two. As shown in figure 5.3 (middle) there exist a clear distinction in the latent space between the two tasks. On the left graph, each episode is clustered but there are no clear patterns that relate different episodes with the same tasks. On the right graph, there's no clear evidence of patterns related to actions. This is expected as each task within an episode requires multiple

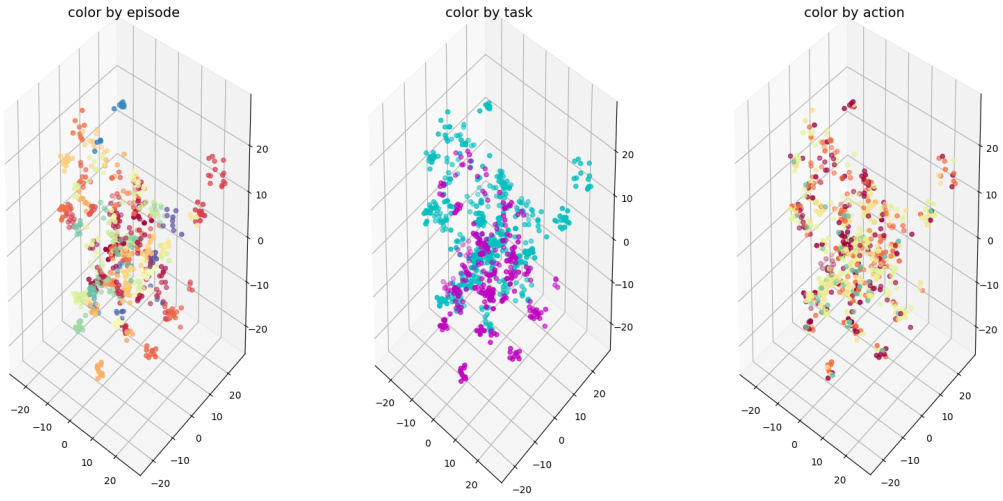
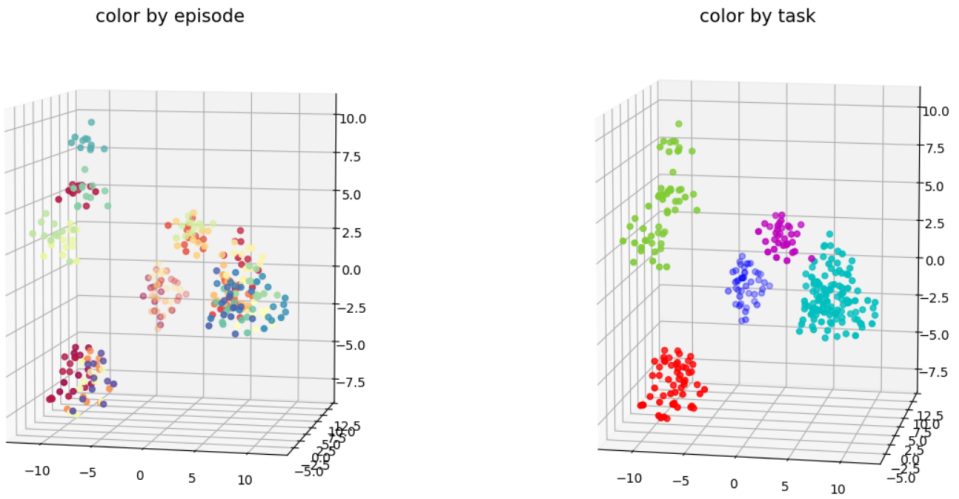


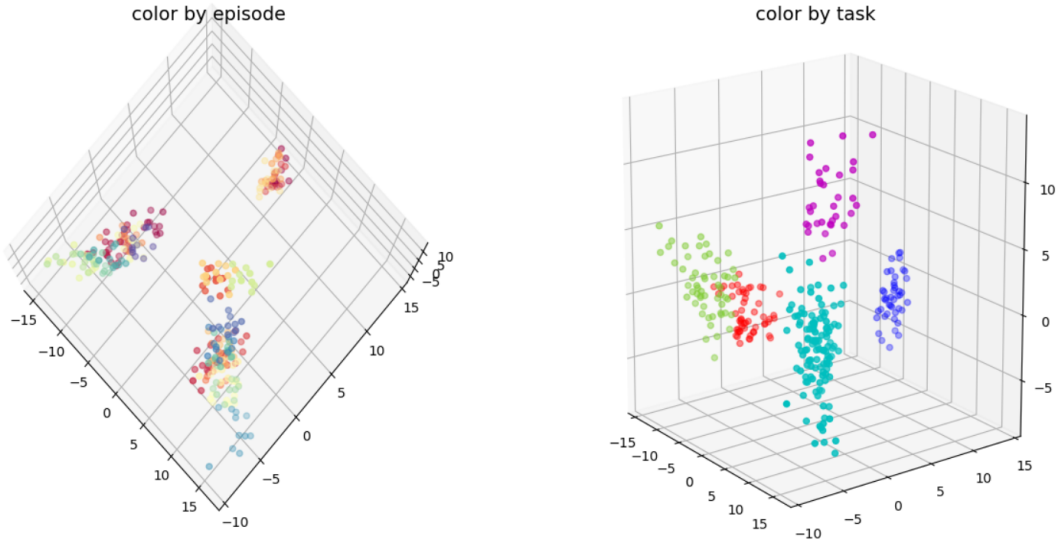
Figure 5.3: We reduced the number of different tasks in the input data and added colour by predicting action on the right. However, there's no clear evidence of patterns related to actions.

different actions to complete. It appears that the plans are grouped more according to the task they solve and the environment they solve it in, while the policy network focuses on translating these into actions.

With this information, we further look at the latent subdomains of the previous chapter independently to explore how they behave in the latent space. As shown in figure 5.4, the pointclouds produced by both the skill and knowledge subdomains show a much clearer cluster pattern when coloured by task. In contrast, both latent subdomains display very little clustering when coloured by episode in graph 5.4a. It is perhaps surprising that the pointcloud produced by the knowledge latent still has a cluster pattern when coloured by task. It is perhaps equally surprising that the knowledge subdomain does not cluster more clearly based on episodes. It is likely that the reason behind this is that episodes solving a given task necessarily include the environmental factors required for that task. For example, we would not attempt to solve the "*Build House*" task in an environment that contains no wood or tools. As such the task to be solved manifests in the environmental context. Similarly, over the course of a single episode, the environmental context changes. Trees get chopped down and tools get moved. We would not necessarily expect the knowledge embedding at the end of an episode to be similar to that at the beginning because the environment has changed.



(a) Skill latent only



(b) Knowledge latent only

Figure 5.4: When visualizing only the skill or knowledge latent in the latent space, the skill latent shows a more tightly clustered pointcloud compared to the knowledge embedding.

These experiments have once again highlighted the strong but incomplete disentanglement of the skill and knowledge within our system. Nevertheless, the clustering observed in many of the experiments supports our hypothesis that this latent space is being regularised effectively and is learning a true compositional representation. In the next section, we will attempt to strengthen the compositionality of our learned representation by treating the regularization of the latent space as an optimization process on the representation embedding network.

5.2.2 Latent Space Regularization

With the information from the previous subsection, we can confirm there exists a possible compositional relationship between vectors in the latent space. Combining this with our work from chapter 4, we can further extend the vector arithmetic in latent space. Previously we required that the embedding for the entire task sequence from step 0 to T must be equal to the sum of the embeddings from the start to the current point $(0...t)$ and from the current point to the end $(t...T)$. Taking this further, any task embedding $\vec{u}$, is a combination of any number of subtasks $\vec{v} \in \vec{u}$ which can themselves be described as an embedding between two-time steps $\vec{v}_{a:b}$. As such, the entire task embedding is a summation of subtask embeddings between various time steps.

$$\vec{u}_{0:T} = \vec{v}_{0:a} + \vec{v}_{a:b} + ... + \vec{v}_{x:T} \tag{5.1}$$

We can apply this idea recursively to the subtasks. As such, any task embedding between two states is the summation of embeddings for all combinations of time steps in between. This leads us to define the constraint,

$$\vec{v}_{a:b} = \vec{v}_{a:c} + \vec{v}_{c:b} \qquad \forall c \quad \text{where} \quad a < c < b \tag{5.2}$$

Ideally, all representations generated in the latent space should follow this constraint, and the latent space would be regularized. In this case, it would be easy for the system to generalize to a sequence of tasks through latent summation.

In order to regulate the learning of the task embedding and optimize over the constraints, we constructed a set of linear equations consisting of a constraints matrix C which defines the constraints and a matrix x which includes all the subtask embeddings. For example, with an

embedding matrix

$$x = \begin{bmatrix} \vec{v}_{0:1} \\ \vec{v}_{1:2} \\ \vec{v}_{0:2} \end{bmatrix} \tag{5.3}$$

The corresponding constraints matrix is:

$$C = \begin{bmatrix} 1, 1, -1 \end{bmatrix} \quad \text{such that} \quad Cx = \vec{v}_{a:c} + \vec{v}_{c:b} - \vec{v}_{a:b} = 0 \tag{5.4}$$

We include all subtask embeddings and constrains within these matrices. We can then easily solve these equations, for example using the Moore-Penrose pseudo-inverse of C. The result is a set of refined task embeddings $\vec{v}^*$ which obey our constraints.

$$\vec{v}^* \Rightarrow \min_{\vec{v}} Cx - z \Rightarrow \min_{\vec{v}} \begin{bmatrix} 1, 1, -1, 0, ... \\ ... \\ 0, 0, ...1, 1, -1 \end{bmatrix} \begin{bmatrix} \hat{v}_{0:1} \\ \hat{v}_{1:2} \\ \hat{v}_{0:2} \\ ... \\ \hat{v}_{T-2:T} \end{bmatrix} - z \tag{5.5}$$

where z is a matrix of zeros.

Of course, optimising the embeddings in this manner does not account for the previous unrefined embeddings. This is likely to cause instability in the system, as the embedding may change drastically over time compared to what the policy has been previously trained on. As a result, we also include an initialization regularizer to try and keep the refined embeddings as close as possible to the unrefined embeddings. To achieve this we append an identity matrix to the end of C, and the previous task embeddings to the end of z. This includes an additional series of equations within the optimization, of the form $\vec{v}^*_{0:1} - \vec{v}_{0:1} = 0$.

As shown in figure 5.5, the optimized pointcloud shown in green has an almost uniform distribution in the latent space. These comprise three square-shaped clusters with a denser distribution on the "edges". Additionally, as shown in figure 5.6, both the small clusters (circled in blue and yellow) outside the main cluster (shown in red) also have their corresponding optimized latent (shown in green) forming clusters outside the main cluster. This shows the consistency of features being preserved throughout the optimization process.

Unfortunately, despite these promising preliminary results, it became obvious that the computational cost of this approach is a major limitation. The number of constraints grows as a factorial

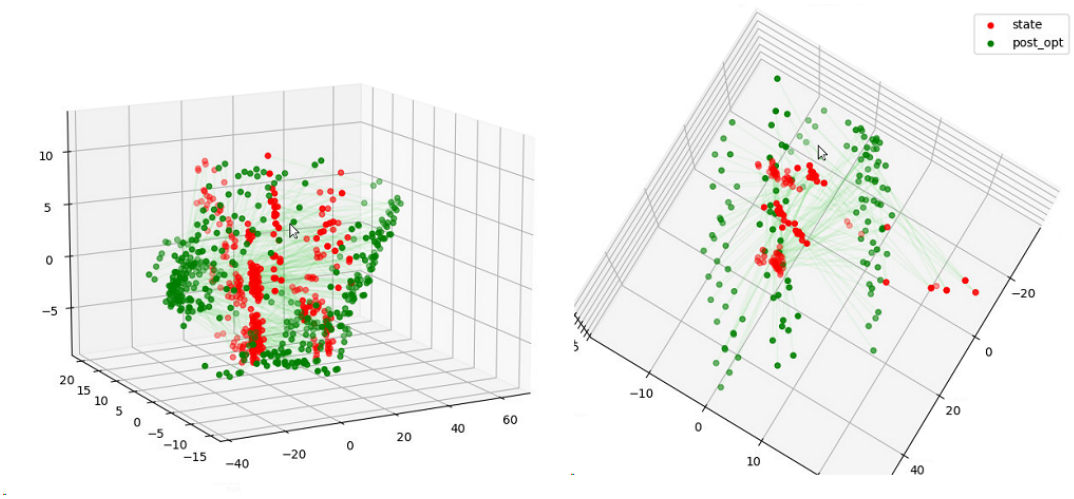


Figure 5.5: Visualization of the original latent and post-optimization latent, the optimized latent shows a greater level of variety and more regular spread in the latent space.

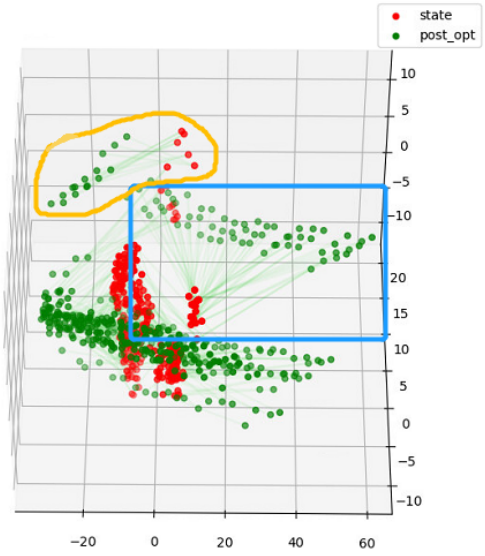


Figure 5.6: The characteristic of the original feature distribution is preserved during the optimization.

to the number of states within a trajectory. As the number of constraint equations increases, so does the size of the constraint matrix and the optimization problem. This causes the computation cost to be prohibitively expensive, in both memory usage and computing time. Therefore, we needed to find a less complex mechanism for enforcing the same type of constraint. The most obvious solution is, instead of performing this constraint optimization process at every step for all previous embeddings, instead generate a single subset of embeddings to optimise at each timestep. To this end, we reverse the problem. Instead of enforcing compositionality constraints on the full history of subtasks to date, we consider how to enforce these constraints for future tasks.

5.2.3 Compositional Subgoals

Our approach CASE [113] generates novel subgoals on the fly, this avoids the problem with factorial computational growth while emphasising the current and future state of the agent. We treat a single complex task as a sequence of smaller implicit tasks or subtasks. Each of these subtasks still requires multiple steps and some effort to learn. However, the need for long-term planning (and the brittleness of divergences) is alleviated. Importantly, unlike previous approaches, we do not explicitly define a finite set of subtasks with hard boundaries (i.e. "*navigate to kitchen*", "*make coffee*" etc.). Instead, the subtasks can be any small sub-trajectory towards the overall goal (e.g. "*Move 3 meters towards the kitchen*", "*pick up the coffee beans*" etc.) and are generated on-the-fly through compositionality. As such, these sub-goals do not need to correspond to any task defined by the developer, nor do they need to have been previously observed during training.

In the remainder of this chapter's work, the robot learns to produce a compositional representation of the task which supports arithmetic operations. This representation is then used to compute a subgoal waypoint in the near future relative to the agent's current progress in the task. Finally, an Imitation Learning policy is trained, using the learned compositional representation as its state space, and with targets set via the adaptive subgoal estimation. These subgoals are adaptive as the rollout progresses, providing additional flexibility. Unlike traditional rigid and non-overlapping subgoals, our approach enables the agent to adapt to errors and drift, following alternative routes to the overall task, and avoiding deadlock with unachievable subtasks. In

addition, the learning task is simplified as long-term planning happens via the compositional space and the agent focuses on short-term execution. This also allows the agent to perform one-shot generalization over unseen tasks in the same environment. With this approach, we are able to outperform standard imitation learning policies (including those using the same compositional state space) by over 30% in unseen task generalization, while maintaining the same level of performance on seen tasks.

In summary, the contributions of this work are the following:

1. A novel approach to estimate subgoal waypoints via a compositional task embedding space
2. An Imitation Learning approach for complex compound tasks, based on online-subgoal estimation
3. An evaluation of one-shot task generalization for the policy, based on subgoal generalization

5.3 METHODOLOGY

In this section, we detail our method of utilizing adaptive subgoal waypoints for learning complex tasks and generalising to previously unseen tasks. In order to define the work in this chapter, we first revisit and formalise the definitions from the introduction. As described in Chapter 1, a "task" (τ) is defined as a singular goal the agent must complete through a series of interactions with the environment. A "sequence"($[\tau_0, \tau_1 ... \tau_n]$) of tasks means a collection of multiple separate tasks, some of which may be independent and some of which may depend on each other. Disregarding task dependencies, we allow the individual tasks within a sequence to be completed in any order, during the completion of the sequence. We further specify a "complex task"($\mathcal{T}$), as a singular specified goal task, which nevertheless engenders an entire sequence of implicitly defined subtasks to be completed ($\mathcal{T} \subseteq [\tau_0, \tau_1 ... \tau_n]$), due to the implicit dependencies within the environment. In our imitation learning framework, we define the "subgoal waypoint" as a state in the expert reference trajectory, located in the "near future" of the current agent's state. Note that the current trajectory and reference trajectory are both solving the

same sequence of tasks, but are operating in different environments. Thus the subgoal waypoint cannot be used directly to guide the agent's trajectory.

As in the previous chapter, we learn a compositional latent space to represent tasks and sequences of tasks. More specifically, a singular task maps to a unique point in the latent space. An unordered sequence of tasks (as defined above) also maps to a unique point in the latent space, which is the summation of the embeddings of all the subtasks within the sequence. This helps to draw a connection between "complex tasks" and "task sequences" as defined above. Both the singular complex task and the explicit sequence of all dependent subtasks should map to the same point within the latent space. This compositional approach makes manifest the lack of ordering specified above. The summation of subtask embeddings is a commutative operation, therefore changing the order of the summation does not change the final embedding. The task embedding learns the representation for a composition of a sequence of tasks and obtains policies that can generalise to a new sequence of tasks. In order to learn this compositional task embedding, this constraint is codified as a number of regularization losses on the state encoder. The regularization losses are the same as in the previous chapter. One loss regulates the summation of the past and future tasks of the system, ensuring it is equal to the embedding of the entire task. The other loss utilizes a marginal loss function to ensure the uniqueness of each training and reference pair, and prevent negative learning.

Finally, we train agents to select actions using the learned task embedding, as their state representation. As in the previous chapter we define O_t as the observation of a state at time t in a fully observable environment. We similarly specify $O^{ref}_{0:T}$ as an expert reference trajectory which completes the target task sequence in a different environment. Therefore, the policy function is:

$$\pi\left(a_t|O_t, O^{ref}_{0:T}, O_{0:t}\right) \tag{5.6}$$

where $O_{0:t}$ is the observation of the episode so far. As in the previous chapter, the expert reference trajectories are extracted by a greedy search over the environment for the optimal solution.

5.3.1 Compositional representation

A compositional representation is an embedding which encodes structural relationships between the items in the space [73]. The work of CPV[33] provided an efficient method for the repre-

sentation of a sequence of tasks. This compositional representation allows the agent to operate on an embedding of the tasks remaining to be done, without ever explicitly defining the target end-state. Consider a compositional representation $\vec{v}$ which consists of the trajectory from state O_0 to state O_T, with an encoding function g_ϕ:

$$\vec{v}_{0,T} = g_\phi(O_0, O_t) = g_\phi(O_0, O_1) + g_\phi(O_1, O_2) + ... + g_\phi(O_{T-1}, O_T) \tag{5.7}$$

This representation defines a sequence of tasks as the sum of the representation for all subtasks within the sequence. To prevent accidentally enforcing a specific ordering during the completion of these subtasks, the representation is built with commutativity, i.e. $\vec{v}_{A,B} + \vec{v}_{C,D} = \vec{v}_{C,D} + \vec{v}_{A,B}$ However, in a long and complex task sequence, the $\vec{v}$ often embeds a long trajectory which consists of multiple tasks later in the sequence. This makes the learning process difficult as the agent will obtain unhelpful information as to its current task.

5.3.2 Plan Arithmetic and Subgoal Waypoints

In one-shot imitation learning, the agent must perform a task (or sequence of tasks) conditioned on one reference example of the same task. In our work, we further generalize this by allowing the current and reference task to be performed under different environments. The agent is trained with many sequences of other tasks in other environments and then provided with an expert trajectory as a reference to guide the new task, with no additional learning. Humans are adept at this: generalizing previous experiences to newly defined problems. However, for machine learning, this is extremely challenging and represents an important stepping stone towards general AI.

During training, the agent is given two trajectories, the training trajectory O and expert trajectory O^{ref} with matching task lists. It then learns a policy to perform online prediction of the actions in one trajectory, conditioned on the other trajectory as the reference. In the running example 'getting coffee', the agent will be provided with trajectories of retrieving coffee from a different office with a different floor layout. Learning how to make coffee without relying on specific meta-knowledge about a particular environment is vital for improving generalization. In imitation learning, the agent is provided with an expert trajectory, which performs the same sequence of tasks at an optimal level.

As in all previous chapters, a visual approach to task specification is taken. During both training and testing, the agent is given an image of the desired goal state of the current episode (O_T), as well as the goal state of the reference episode ($O^{ref}_{0:T}$). It is also given an image of the current state (O_t), and an image of a future subgoal state from the reference trajectory (O^{ref}_I). It is important to emphasise that the agent is not provided with any future knowledge about the current trajectory, beyond the target goal state which is used to specify the task to be completed. Subgoals are drawn from the future of the reference trajectory, not the current trajectory ($O^{ref}_I \in O^{ref}_{0:T}$).

The model will first encode both the compositional representation of the current state to the goal state ($O_{t:T}$), and the compositional representation of the reference sub-goal to the goal state of the reference episode($O^{ref}_{I:T}$). It will then use the difference between the two ($O^{ref}_{I:T}$ - $O_{t:T}$) to predict the next action. As in the last chapter, Let $\vec{u}_{0,T} = g_\phi(O_{a:b})$ embed the observation pair at time a and b into the compositional representation with encoder g and parameters ϕ. We can estimate a subgoal state O^{ref}_I within the reference trajectory $\{O^{ref}_0, O^{ref}_T\}$, and create an compositional representation from this waypoint state to the goal state of the expert trajectory $\vec{v} = g_\phi(\{O^{ref}_I, O^{ref}_T\})$. Let $\vec{u} = g_\phi(\{O_t, O_T\})$ be the representation from the current state to the goal state, then we can calculate a waypoint representation $\vec{W}$ with the following subtraction in the latent domain:

$$\vec{W} = \vec{u} - \vec{v} = g_\phi(\{O_t, O_T\}) - g_\phi(\{O^{ref}_I, O^{ref}_T\}) \tag{5.8}$$

At timestep t, equation 5.8 estimates an approximation $\vec{W} = g_\phi(\{O_t, O^{ref}_I\})$ of the trajectory from the current state of the agent to the subgoal waypoint without having to explicitly know the waypoint along the current trajectory. This representation is then used as input for policy network $\pi\left(a_t|O_t, \vec{W}\right)$ to determine the actions of the agent.

To choose the subgoal waypoint, we assume the agent is always on the optimal path, therefore its progress in the task is proportional to that of the expert trajectory. As such when we choose the waypoint, we take the state O^{ref}_p in the reference trajectory, which has the same percentage of completion as in the training episode ($\{O^{ref}_0, O^{ref}_p\}/\{O^{ref}_0, O^{ref}_T\} = \{O_0, O_t\}/\{O_0, O_T\}$), then add a fixed subgoal lookahead parameter k steps to ensure the waypoint is in the near future. For example, in a training episode, the agent must perform two tasks: *Chop Tree* and *Build House*. In the reference trajectory, *Chop Tree* takes 10 steps and building a house takes 5 steps.

In the training trajectory, the chop tree task has 15 steps, and the build house takes 15 steps. An agent at the 6th step in the training trajectory is considered to have the same percentage of completion as the 3rd step in the reference trajectory. Then we apply a $k = 4$ step lookahead, the 7th step in the reference trajectory is now the subgoal waypoint. An apparent issue with this approach is the problem of overreaching. When an agent is at the 14th step in the training trajectory, the new subgoal waypoint is at the 11th step in the reference trajectory, which falls into the next task. This will result in a misleading demonstration and potentially confuse the agent in the current task. However, this issue can be avoided by carefully picking the parameter k.

The value of k is determined experimentally in our work, as in previous works in the broader subgoal selection literature [29]. Their work concluded a higher k should make planning easier as this lowered the search graph for future actions. This comes at the cost of the quality of the subgoal generator, and uncertainty in overall performance. As k increases in an episode, the chance of the subgoal state R_I landing in a different task in the reference sequence increases. The new task in the reference sequence is likely not an ideal demonstration for the current task in the training sequence. The optimal value of k varies depending on the tasks and the working environment as well as the subgoal system applied for learning. One potential issue with this approach is that the length of each subtask is unknown. If the current subtask in the training episode is significantly longer or shorter than the expert trajectory, then the waypoint may fall into a different subtask. However, we expect the agent to be able to adapt to this situation, as any state from the following subtask will already reflect the completion of the current subtask.

Based on our new definition of the subgoal policy, the action loss becomes:

$$L_a(O_t, O_I^{ref}) = -log\Big(\pi\Big(\hat{a}_t|O_t, g_\phi(\{O_t, O_T\}) - g_\phi(\{O_I^{ref}, O_T^{ref}\}\Big)\Big) \tag{5.9}$$

Additionally, the two compositional regularization losses from the previous chapter are still used. L_H from eq 4.4 enforces the compositionality within the current trajectory while L_P from eq 4.5 ensures that similar tasks are encoded into similar regions of the latent space.

5.4 Evaluation

Our experiments aim to show the improvement of the agent's combinatorial generalization capability when trained using online compositional subgoals. Therefore, we evaluate the performance on tasks which were previously unseen during training, and for which only a single reference episode is provided. In these new tasks, the reference episode is also performing unseen tasks in an unseen environment. In both cases, the observation is provided as a pair of images. The state encoder is a 4-layer CNN and is shared by the reference and training episode. This encodes the current state to the goal state sub-trajectory, as well as the reference sub-goal state to the goal state sub-trajectory. The resulting latent will then be processed according to equation 5.8, and the output is fed into the policy network to estimate the action. Note that for clarity, these results do not include the gated VAE approach with disentangled skill and knowledge of the previous chapter. This allows us to focus on evaluating the advantages of training with adaptive subgoals.

In each experiment, we contrast several variants of our own approach, including the effect of the current image branch and the additional compositionality losses. We also compare against the current state-of-the-art in compositional IL [33]. Additionally, we include an ablation study on the "near future" subgoal lookahead parameter k. In all other experiments, we set $k = 4$ due to the results of the ablation study.

5.4.1 Environment

In order to examine generalization and long-term IL problems, we trained our agent in the Craft World[32] environment. This is the same environment we used in the previous chapter. This also facilitated a fair comparison with the state-of-the-art compositional IL approach [33]. This environment is a 2D discrete-action world with a top-down grid view where the agent is also able to move in one of 4 directions at each step. There are different types of objects in the environment as shown in figure 5.7 including trees, rocks, axe, wheat, bread, etc. Some objects can be interacted with by an agent via pick-up and drop-off actions. The object moves with the agent when picked up, which can cause transformations to other objects in the environment. For example, if the agent carries an axe to a tree, the tree will be transformed into a log, which then

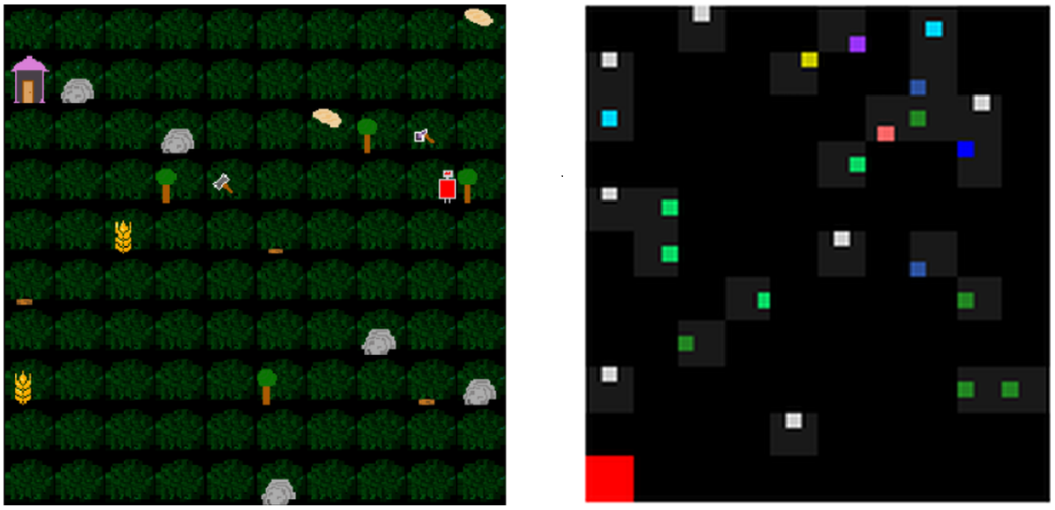


Figure 5.7: An example of the Crafting World[32] environment, on the left, is a rendering of a state observation, and the right image is a different state observation received by the agent.

can be transformed into a house once the agent picks up a hammer and brings it to the log. It is apparent that this environment makes it possible to define complex long-horizon tasks such as "*Make Bread*" or "*Build House*" which include many implicit subgoals. Furthermore, these tasks can be combined into sequences such as ["*Make Bread*", "*Eat Bread*", "*Build House*"]. This provides a good selection of individual tasks to generate training data. This also liberates the agent from skill list labels [81] or language-based skill descriptions [95] which limits the generalization to unseen tasks and sequences. Also of interest is the fact that this allows the task sequence to be generated with no explicit ordering, giving more freedom in both data generation and generalization.

The training and testing dataset is generated with a random map, upon which the agent is required to perform 2-8 different tasks in sequence with no particular ordering. The expert trajectory is generated with a greedy search to ensure an optimal solution. The agent is trained on 150,000 episodes and tested on the same number of unseen episodes. For the standard experiment, the list of training and testing tasks was the same. For the one-shot generalization experiment, the set of training tasks is different from the set of testing tasks, requiring generalization from the reference trajectory during training.

5.4.2 One-shot task generalization

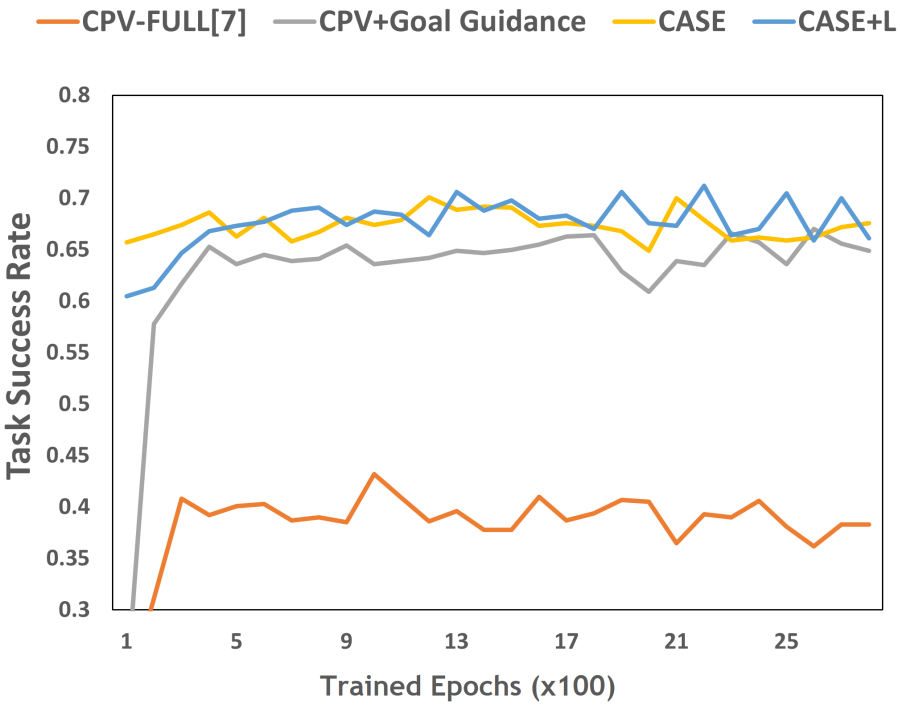


Figure 5.8: This graph shows the unseen task success rate over epochs of our subgoal agent with and without the regularization losses and a modified SOTA benchmark. The original CPV-FULL[33] has a much lower performance overall, while the modified version has a closer performance compared to our agent after 500 epochs. The inclusion of regularization losses (CASE+L) only improved the agent when performing seen tasks, the improvement in unseen task generalization is limited.

During the generalization test, we train the agent for 3000 epochs and test the agent's ability to perform in a complete unseen environment with unseen tasks. As this work borrows the idea of compositional plan vector, we again use the work of CPV [33] as the SOTA benchmark. Results from the generalization test are shown in figure 5.8. Our agent is able to outperform the original SOTA benchmark [33] by over 30% in unseen task success rate consistently throughout the training process. The SOTA model requires both a complete reference trajectory and the trajectory so far, to calculate the future plan for the agent. However, in its original form [33] does not take the desired goal state as input at test time. Instead, the approach relies on the

compositionality of the current and reference trajectory to produce a goal state. To ensure a fair comparison, we also evaluate an enhanced version of [33] (noted as CPV+Goal Guidance), where the reference trajectory is replaced with the ground truth end-goal of the current trajectory at test time. This model shows a higher performance after 500 epochs of training.

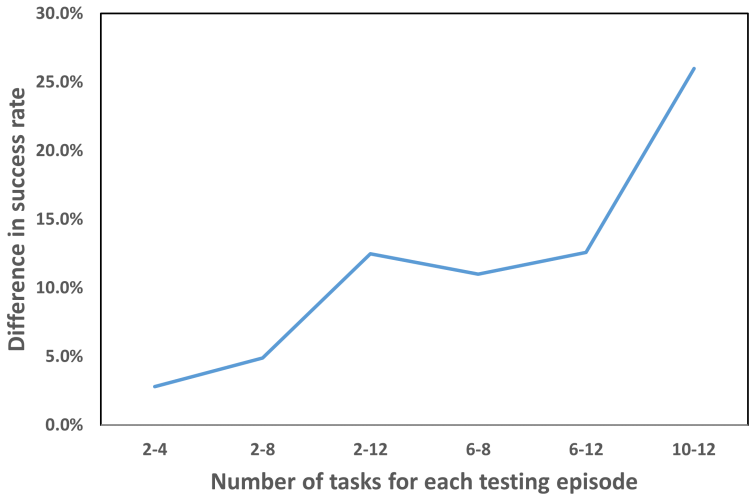


Figure 5.9: The difference in performance increases as the number of tasks in a sequence increases.

We also performed the evaluation previously described in Chapter 4.3.4, where performance is evaluated against sequence length. We can observe a growing difference in performance as the number of tasks in a sequence increases as shown in figure 5.9. When the number of tasks in each episode sequence is chosen from the range 2-4, the proposed approach outperforms the baseline by around 3-4%. When sequence lengths are drawn from the range 2-8, this gap roughly doubles. Our model is able to maintain an overall success rate above 50% throughout the experiments, this is considered a successful one-shot task generalization.

5.4.3 Ablation Study

Next, we perform an ablation study of the different components of the framework. More specifically we explore the benefits of the regularization losses (L_H from eq 4.4 and L_P from eq 4.5), and the current state image input (O_t). As shown in table 5.1, the addition of both the current state image branch and the regularization losses each improved the performance

of the model. As expected, the current image provided additional information more relevant to the current task, while the regularization losses increased the generalization capability by encouraging a more regularized representation.

Model	Best Performance	Average Performance	Standard Deviation
CPV-FULL[33]	0.432	0.392	0.0166
CPV+Goal Guidance	0.670	0.647	0.0146
CASE	0.689	0.641	0.0133
CASE+CI	0.701	0.676	0.0139
CASE+CI+L_R	**0.712**	**0.687**	0.0167

Table 5.1: An ablation study on the different components of the network. Current state image (CI) and assistive losses ($L_R = L_H + L_P$) show some improvement in generalization capability.

5.4.4 Hyperparameter Search

Finally, we tested several settings for the "near future" lookahead parameter k. As shown in figure 5.10, when $k = 4$ the agent's performance is maximized. The graph shows some sensitivity to the parameter k, and performance can be unstable at lower values. We suspect this is due to the inconsistency in the length of the randomized subtasks between the training episodes and expert trajectories. As an example, imagine the first task in the training episode is to pick up an axe. In the agent's training episode the axe is maybe 10 steps away from the current state, while it is 2 steps away in the expert reference trajectory. If the k value is less than 2, then the generated subgoal will be after the completion of the current subtask within the expert trajectory. The reverse can also be true for larger values of k if the distances are swapped, where the reference trajectory subgoal points towards a task that the agent has already completed. In most cases, the agent is able to deal with this: a subgoal for the following task is still easier to learn from than the entire remaining trajectory. Nevertheless, it may be interesting for future work to explore the automatic computation of the optimal k parameter during compositional subgoal estimation.

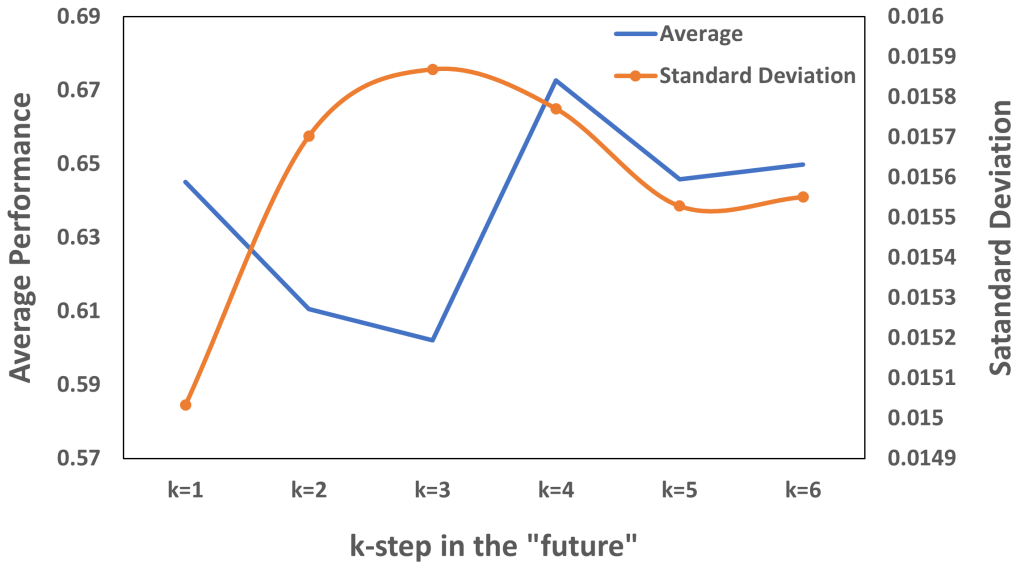


Figure 5.10: An ablation study over the number of k-step for choosing the near future waypoint. The average performance of each k-step is calculated from 8 trials. The setting $k = 1$ has mediocre performance but the lowest standard deviation, $k = 3$ has the opposite result. We used $k = 4$ for our experiments as this setting has the highest performance.

5.5 Conclusion

In this work, we proposed to learn a compositional task representation which enabled novel subgoal estimation from reference trajectories in IL. This makes it significantly easier to learn long and complex sequences of tasks, including those with implicit or poorly defined subtasks. It also enabled one-shot task generalization where the agent can solve tasks it has never seen during training, using only a single reference trajectory in a different environment. With this technique, we developed an IL agent which can generalize to previously unseen tasks with a success rate of around 70%. This represents an improvement of around 30% over the previous SOTA.

However, this approach can be developed further in future work. As discussed in section 5.4.3, using a fixed value for the k-step lookahead parameter is likely suboptimal. It is likely that performance and stability may be improved by developing an adaptive lookahead window, based on recent developments in the broader field of subgoal search [29].

Additionally, the training and testing tasks we used in this work are all navigation related and performed in a toy environment. We have no evidence that this approach will achieve similar improvement for more complex tasks. From previous works, we know the subgoal method also has good performance in other area such as classification, manipulation and motion control. Expanding this work to other types of tasks could be a potential benchmark for further testing.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The future of robotics calls for increased multi-tasking capabilities in order to effectively generalize across a wide range of tasks in diverse and unconstrained environments. The motivation behind the research conducted in this PhD is rooted in the recognition that robots will require adaptive skills to successfully perform these tasks in various locations, guided by an AI-based high-level planner.

Emulating the human ability to adapt to new environments poses a significant challenge for the fields of robotics and artificial intelligence. The aim is to develop robots that can seamlessly adjust their behavior and skills to suit the demands of unfamiliar settings, mirroring the adaptability demonstrated by humans. As such, the main objectives for the thesis were:

1. To develop a robotic system which utilized a reinforcement learning multi-environment planner and a traditional navigation method for low-level control.

2. To develop a multi-task learning framework which learns a disentangled representation of robotic tasks, with experience sharing and generalization capability.

3. To explore the reinforcement and imitation learning solutions for long and complex tasks, utilizing the idea of sub-tasks and combinatorial generalization.

This chapter will summarize how these objectives have been achieved and tested in both simulation and real-world environments. By achieving these objectives, this thesis contributed to the state of the art.

In Chapter 3 of this thesis, we introduced the concept of multi-environment navigation. This enabled an AI-controlled robotic navigation framework which was based on multi-task deep reinforcement learning. We demonstrated our work with the problem of visual navigation in both simulated and real-world environments. This work combined techniques from multi-task learning, reinforcement learning and visual navigation. The result took aspects from each to formulate a more deployable robotic system. By utilizing attentional task allocation and environment type classification, we enabled the agent to navigate in multiple environments within one training session while sharing learning between environments. Overall, the MERLIN model outperformed the SOTA visual navigation model in both performance and training speed for the multi-environment setting, as well as the single-environment setting. We observed different behaviours depending on the surroundings in both the simulated and real-world environments. We also demonstrated the model's ability to transfer to the real world after training offline in a discrete simulation environment. This work satisfied the first objective of the thesis.

In the second technical chapter, we approached the problem of multi-task learning from a neurobiological perspective. This work achieved the second objective of the thesis. Taking inspiration from neurobiology and pedagogy studies on memory acquisition, we hypothesized the latent space in a policy neural network could be disentangled into subdomains. The subdomains each contain the learned information for either the skill or the knowledge of a task. These should be independently transferable to solve unseen tasks. We successfully demonstrated this disentanglement in imitation learning, using a gated VAE architecture. With our method, we were able to outperform the SOTA model, in two different environments, both in terms of success rate and speed. During our experiments, we discovered that skill and knowledge can not be completely disentangled, this finding echoes the studies in pedagogy and language acquisition. Even in humans, certain elements of procedural memory are entangled with declarative memory. This disentanglement of skill and knowledge was a fundamental step toward combinatorial generalization. Combing this chapter with the work in Chapter 3, we explored the generalization aspect of machine learning.

In the last technical work presented in Chapter 5 of this thesis, we proposed a compositional task representation learning framework. This work achieved the third objective of the thesis. The CASE framework utilized a novel subgoal estimation approach from reference trajectories. This reduced the cost of learning complex task sequences within an imitation learning setting. These "near future" subgoals were recalculated at each step using compositional arithmetic in learned latent representation space. This made it significantly easier to learn long and complex sequences of tasks, including those with implicit or poorly defined subtasks. In addition to improving learning efficiency for standard long-term tasks, this approach also made it possible to perform one-shot generalization to previously unseen tasks, given only a single reference trajectory for the task in a different environment.

Overall, this thesis presents several novel solutions for combinatorial generalization. By utilizing reinforcement and imitation learning in a multi-task learning framework, we demonstrated skill and knowledge recombination, and long-term reasoning tasks across multiple environments. This has led to 3 publications at ICRA and IROS based on the work in Chapter 3 [114], Chapter 4 [115] and Chapter 5 [113] respectively. This thesis shows that a better understanding of human behaviours can provide valuable insight into generalizable machine learning. The techniques presented in this thesis show several clear directions for future work.

6.2 Limitations and Short-Term Future Work

One potential area for future work is to explore the integration and unification of the various techniques presented in the thesis. While combining these contributions may require significant development, it has the potential to create a valuable benchmark for future reinforcement learning and imitation learning systems. By investigating the synergies and interactions between the different approaches, it may be possible to develop a more comprehensive and robust system.

We will next consider possible extensions to the chapters independently. During the experiments in Chapter 3, we find the optimal number of sub-networks is not directly proportional to the number of environments during training. In fact, the optimal number of sub-networks is often lower. We suspect this is due to the transferring of skill and knowledge between each navigation task, as similar behaviour can be observed across similar types of environments. Therefore,

it would be interesting to explore letting the network determine the optimal number of sub-networks automatically. By dynamically increasing or decreasing the number of sub-networks during training and testing, the framework could adapt to various tasks with minimal additional resources cost while maintaining performance.

In the real-world application of Chapter 3, the robot demonstrated an ability to auto-correct the overshot in rotation. However, this behaviour often took several attempts to complete. This is likely caused by the limited viewing angles in the training environment. At the time of development, the AI2-Thor environment had limited handling of angular movement. This limited our ability to introduce angular change in the training. During the noise resilience experiments, we did not introduce any angular noise in the system, the shifted samples are all taken from the same angular view as the original state. This minor limitation caused the robotic system to perform slightly less reliable than its potential. The recovery movement often took more steps to complete than the actual task.

In Chapter 5 the look-ahead parameter k was determined experimentally during this work. While previous research [29] also followed this approach and results were favourable, it may be possible to develop a better method for determining an optimal k. We could instead explore a learned approach where the optimal k parameter is estimated on the fly, based on the current conditions. With this method, the network can better avoid the overlapping issue during training, and potentially achieve a better performance overall.

6.3 Directions for the Field

In the long term, the contributions of this thesis pave the way for advancements in combinatorial generalization in machine learning. It has been recognized that the ability to generalize beyond learned experiences remains a significant challenge in AI research, as highlighted in the work of Battaglia et al.[10]. Combinatorial generalization, particularly with relational inductive biases embedded into the framework, will be a crucial building block for AI agents to achieve human-like capabilities.

Combinatorial generalization refers to the unique capacity of human intelligence to combine and construct new inferences, predictions, and behaviours from known experience building

blocks. Combinatorial generalization refers to the signature ability of human intelligence which is able to combine and construct new inferences, predictions and behaviours from known experience building blocks. Early research has concluded that a human's capacity for Combinatorial generalization is based on the cognitive ability to represent structure and reason about relationships between entities[49, 35]. This enables humans to solve novel problems by inferring the relation between the new problem and our known abstract hierarchy. We can then pick and choose the combination of known skills and knowledge to solve the problem. This ability to make analogies by comparing and contrasting relational structures is critical for both combinatorial generalization and domain adaptation. The work presented in Chapter 3 and Chapter 5 offers an opportunity to embed this type of structure. More specifically, a relational graph of expert networks from chapter 3 may help build an abstract knowledge graph within the compositional latent space of chapters 4 and 5. This graph should automatically adapt and grow when exposed to new problems.

A general direction for future work is closing the simulation gap and making the training environment and setting more realistic. The difficulty in transitioning between training and real-world environments has been well documented. Although we have had some success within this thesis, broader applications in the real world are likely to be challenging. Unfortunately, it is likely that more advanced simulators will require greatly increased computing resources and would adversely affect training speed. This issue has been the major challenge in reinforcement learning research being deployed in the real-world. The nature of reinforcement learning demands interaction with the environment during training, and this is difficult and costly for a real world environment. Most of the time learning a successful policy in simulation does not transfer to real-world deployment, especially for fields such as robotics[28]. More recent research on this topic often focuses on the simulation side, making the simulation more dynamic and randomized [84], or more detailed and synthesized to mimic reality [15]. Our work in Chapter 3 took the approach of making the simulation using a real-world image, however, this is costly and inaccurate. Even better simulations take the physical properties of the system into account [5]. However, this on only half of the solution, making the simulation mimic reality more accurately is very costly in many aspects, especially in computational resources and training time. It is possible to develop a universal training framework which enables the agent to achieve domain adaptation, especially with robotic systems.

Another vital step towards a true general AI is the development of a universal agent. This would be a single agent which can simultaneously learn to solve a large number of disparate tasks, ideally sharing learning between these tasks. The closest prior work is [85] but this work learned to solve each task separately through isolated training. One avenue for exploring a truly general agent is learning within modern video games. There have been several recent examples where video games are used as training environments for research [112, 38, 23]. However, these researches almost always focus entirely on a single game or a single environment within the game, and common skills are not transferred between games. A truly general agent would be able to generalize over many games, and be able to share related skills like navigation and problem solving, despite the different abstractions for these skills present in each environment. The biggest challenge is to consider how skills and knowledge can be effectively shared between environments with vastly different observation and action spaces. Only the most abstract aspects of these skills can be shared.

Overall, we hope that these paths for future work will all bring us closer to an AI agent which can solve multiple problems and adapt well to new tasks, when deployed on real-life robotic hardware. This would surely usher in a new age of AI and robotics.

Bibliography

[1] Multi-task learning for HIV therapy screening | proceedings of the 25th international conference on machine learning.

[2] Robin Amsters and Peter Slaets. Turtlebot 3 as a robotics education platform. In *International Conference on Robotics in Education (RiE)*, pages 170–181. Springer.

[3] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR.

[4] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems(NIPS)*.

[5] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[6] Abdul Fatir Ansari and Harold Soh. Hyperprior induced unsupervised disentanglement of latent representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3175–3182.

[7] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR.

[8] Shi Bai, Fanfei Chen, and Brendan Englot. Toward autonomous mapping and exploration for mobile robots through deep supervised learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2379–2384. IEEE.

[9] Bram Bakker, Jürgen Schmidhuber, et al. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*, pages 438–445. Citeseer.

[10] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[11] Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[12] Samy Bengio, HM Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Advances in neural information processing systems 31. In *Annual conference on neural information processing systems 2018, NeurIPS 2018, 3–8 December 2018*. Montréal Canada.

[13] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. 19(5):1179–1187. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[14] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. 7(3):278–288.

[15] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[16] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Model-free reinforcement learning for stochastic games with linear temporal logic objectives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10649–10655. IEEE, 2021.

[17] Timo Bräm, Gino Brunner, Oliver Richter, and Roger Wattenhofer. Attentive multi-task deep reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 134–149. Springer.

[18] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae.

[19] Mark Burgin. *Theory of Knowledge: Structures and Processes*, volume 5. World scientific.

[20] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pages 941–950. PMLR, 2019.

[21] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1430–1440. PMLR.

[22] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. 33:4247–4258.

[23] Devendra Singh Chaplot and Guillaume Lample. Arnold: An autonomous agent to play fps games. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] Kevin Chen, Junshen K. Chen, Jo Chuang, Marynel Vazquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11276–11286.

[25] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders.

[26] Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast policy learning through imitation and reinforcement.

[27] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. 34:1–25.

[28] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[29] Konrad Czechowski, Tomasz Odrzygóźdź, Marek Zbysiński, Michał Zawalski, Krzysztof Olejnik, Yuhuai Wu, Łukasz Kuciński, and Piotr Miłoś. Subgoal search for complex reasoning tasks. 34:624–638.

[30] Bin Dai and David Wipf. Diagnosing and enhancing vae models.

[31] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. Integrating state representation learning into deep reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1394–1401, 2018.

[32] Coline Devin. craftingworld. original-date: 2019-07-11T16:56:42Z.

[33] Coline M Devin, Daniel Geng, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Compositional plan vectors.

[34] Wei Gao, David Hsu, Wee Sun Lee, Shengmei Shen, and Karthikk Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. In *Conference on Robot Learning*, pages 185–194. PMLR.

[35] Geoffrey P Goodwin and PN Johnson-Laird. Reasoning about relations. *Psychological review*, 112(2):468, 2005.

[36] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling.

In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE.

[37] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR.

[38] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

[39] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[40] Masayoshi Hashima, Fumi Hasegawa, Shinji Kanda, Tsugito Maruyama, and Takashi Uchiyama. Localization and obstacle detection for robots for carrying food trays. In *1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 345–351. IEEE.

[41] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[42] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework.

[43] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization.

[44] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. 29.

[45] Berthold KP Horn and Brian G Schunck. Determining optical flow. 17(1-3):185–203.

[46] Chuan-En Hsu, Mahdin Rohmatillah, and Jen-Tzung Chien. Multitask generative adversarial imitation learning for multi-domain dialogue system. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 954–961. IEEE, 2021.

[47] Keishi Ishihara, Anssi Kanervisto, Jun Miura, and Ville Hautamaki. Multi-task learning with attention for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2902–2911.

[48] Mansur Kabuka and A Arenas. Position verification of a mobile robot using standard pattern. 3(6):505–516.

[49] Charles Kemp and Joshua B Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008.

[50] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2938–2946.

[51] Dongsung Kim and Ramakant Nevatia. Symbolic navigation with a generic map. 6(1):69–88.

[52] Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. 34.

[53] Ye-Hoon Kim, Jun-Ik Jang, and Sojung Yun. End-to-end deep learning for autonomous navigation of mobile robot. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE.

[54] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

[55] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai.

[56] Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi, and Joshua B Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Neural Information Processing Systems(NIPS)*.

[57] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations.

[58] Nicholas C Landolfi, Garrett Thomas, and Tengyu Ma. A model-based approach for sample-efficient multi-task reinforcement learning.

[59] Przemyslaw A Lasota, Gregory F Rossano, and Julie A Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 339–344. IEEE.

[60] Seunghak Lee, Jun Zhu, and Eric P Xing. Adaptive multi-task lasso: with application to eqtl detection. In *Advances in neural information processing systems*, pages 1306–1314.

[61] Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR.

[62] Siyuan Li, Jin Zhang, Jianhao Wang, Yang Yu, and Chongjie Zhang. Active hierarchical exploration with stable subgoal representation learning.

[63] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael Bowling. State of the art control of atari games using shallow reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 485–493. International Foundation for Autonomous Agents and Multiagent Systems. event-place: Singapore, Singapore.

[64] Bo Liu, Xuesu Xiao, and Peter Stone. A lifelong learning approach to mobile robot navigation. 6(2):1090–1096.

[65] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l2, 1-norm minimization.

[66] Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning. In *International Conference on Machine Learning*, pages 6259–6270. PMLR.

[67] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880.

[68] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Imaging Understanding Workshop*, pages 121–130. Vancouver, British Columbia.

[69] Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444. IEEE, 2022.

[70] Bar Mayo, Tamir Hazan, and Ayellet Tal. Visual navigation with spatial attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16898–16907.

[71] Robert McCormick. Conceptual and procedural knowledge. 7(1):141–159.

[72] Oscar Mendez, Simon Hadfield, and Richard Bowden. Markov localisation using heatmap regression and deep convolutional odometry. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644. IEEE.

[73] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. 26.

[74] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

[75] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. 518(7540):529–533.

[76] Steven D. Morad, Roberto Mecca, Rudra P. K. Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied visual navigation with automatic curriculum learning in real environments. 6(2):683–690.

[77] Hans P Moravec. The stanford cart and the cmu rover. 71(7):872–884.

[78] Adithyavairavan Murali, Animesh Garg, Sanjay Krishnan, Florian T Pokorny, Pieter Abbeel, Trevor Darrell, and Ken Goldberg. Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4150–4157. IEEE.

[79] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. Ieee.

[80] Guillaume Obozinski, Ben Taskar, and Michael Jordan. Multi-task feature selection. 2(2.2):2.

[81] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR.

[82] G. Oriolo, M. Vendittelli, and G. Ulivi. On-line map building and navigation for autonomous mobile robots. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 2900–2906 vol.3. ISSN: 1050-4729.

[83] Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. 32.

[84] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[85] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[86] Bethany Rittle-Johnson and Robert S Siegler. The relation between conceptual and procedural knowledge in learning mathematics: A review.

[87] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks.

[88] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks.

[89] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation.

[90] Stefan Schaal. Is imitation learning the route to humanoid robots? 3(6):233–242.

[91] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[92] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

[93] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms.

[94] Soroush Seifi, Abhishek Jha, and Tinne Tuytelaars. Glimpse-attend-and-explore: Self-attention for active visual exploration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16137–16146.

[95] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning.

[96] David Silver, James Bagnell, and Anthony Stentz. High performance outdoor navigation from overhead data using imitation learning. 1.

[97] Avi Singh, Eric Jang, Alexander Irpan, Daniel Kappler, Murtaza Dalal, Sergey Levinev, Mohi Khansari, and Chelsea Finn. Scalable multi-task imitation learning with autonomous improvement. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2167–2173. IEEE, 2020.

[98] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. 31.

[99] Jaime Spencer, Richard Bowden, and Simon Hadfield. Medusa: Universal feature learning via attentional multitasking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3800–3809.

[100] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning.

[101] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pages 3309–3318. PMLR.

[102] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. 20.

[103] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Neural Information Processing Systems(NIPS)*.

[104] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning.

[105] Michael T Ullman. The declarative/procedural model: a neurobiological model of language learning, knowledge, and use. In *Neurobiology of language*, pages 953–968. Elsevier.

[106] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[107] Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. Gated variational autoencoders: Incorporating weak supervision to encourage disentanglement. In *2020*

15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), pages 125–132. IEEE.

[108] Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. Nestedvae: Isolating common factors via weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9202–9212.

[109] Lu Wang, Ruiming Tang, Xiaofeng He, and Xiuqiang He. Hierarchical imitation learning via subgoal representation learning for dynamic treatment recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1081–1089, 2022.

[110] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[111] Daniel B Willingham, Mary J Nissen, and Peter Bullemer. On the development of procedural knowledge. 15(6):1047.

[112] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[113] Bian Xihan, Oscar Mendez, and Simon Hadfield. Generalizing to new tasks via one-shot compositional subgoals.

[114] Bian Xihan, Oscar Mendez, and Simon Hadfield. Robot in a china shop: Using reinforcement learning for location-specific navigation behaviour. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5959–5965. IEEE.

[115] Bian Xihan, Oscar Mendez, and Simon Hadfield. Skill-il: Disentangling skill and knowledge in multitask imitation learning.

[116] Junhong Xu, Qiwei Liu, Hanging Guo, Aaron Kageza, Saeed AlQarni, and Shaoen Wu. Shared multi-task imitation learning for indoor self-navigation. In *2018 IEEE global communications conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[117] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors.

[118] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[119] Xinyi Yu, Jianan Hu, Yuehai Fan, Wancai Zheng, and Linlin Ou. Multi-subgoal robot navigation in crowds with history information and interactions. *arXiv preprint arXiv:2205.02003*, 2022.

[120] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. 33:21579–21590.

[121] Yu Zhang and Qiang Yang. An overview of multi-task learning. 5(1):30–43.

[122] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning.

[123] Zhilin Zheng and Li Sun. Disentangling latent space for vae by label relevant/irrelevant dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12192–12201.

[124] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE.