LEARNING MORE FROM LIMITED DEMONSTRATIONS: METHODS FOR EFFICIENT AND INFORMATIVE HUMAN-ROBOT INTERACTION

by

QINGHUA CHEN

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2025

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Osamah A. Rawashdeh, Ph.D., Co-Chair
Wing-Yue Geoffrey Louie, Ph.D.,Co-Chair
Jessica Korneder, Ph.D.
Guangzhi Qu, Ph.D.
Yanfeng Wang, Ph.D.

ACKNOWLEDGMENTS

The research in this dissertation is the result of support and help from many individuals and institutions. I want to express my sincere appreciation to all who contributed to this journey, both academically and personally.

First, I would like to thank my advisors, Dr. Osamah A. Rawashdeh and Dr. Wing-Yue Geoffrey Louie, for their valuable guidance and encouragement. Their high academic standards and insightful advice have greatly influenced my research and improved my skills. I am also thankful to Dr. Jessica Korneder for her support and for providing essential data for this work, and to Dr. Yanfeng Wang for his continuous support and encouragement during these years.

I am also grateful to my thesis committee members for their constructive feedback and suggestions, which helped improve and strengthen this dissertation. My thanks also go to my team members—Sean, Motaz, Evan, Dejin, Pourya, Alex, and Alaaldin—for their collaboration, insights, and advice, which improved the quality of this work and broadened my perspective.

Finally, I give my deepest thanks to my family, especially my daughter Chongxi and Mr. J, for their love, support, and encouragement.

Qinghua Chen

ABSTRACT

LEARNING MORE FROM LIMITED HUMAN ROBOT INTERACTION (HRI) DEMONSTRATIONS

by

QINGHUA CHEN

Adviser: Osamah A. Rawashdeh, Ph.D, and Wing-Yue Geoffrey Louie, Ph.D.

Learning from Demonstration (LfD) offers a promising paradigm for enabling socially assistive robots (SARs) to acquire complex skills and social behaviors by observing human demonstrations. However, conventional LfD methods rely heavily on large-scale, high-quality demonstration datasets, which are difficult to obtain in real-world healthcare and education settings due to privacy, cost, and data scarcity. This dissertation addresses these challenges by proposing a series of approaches to enhance learning efficiency, improve adaptability, and maximize information extraction from limited demonstration data in human-robot interaction (HRI) scenarios.

First, a hierarchical deep reinforcement learning framework is introduced that incorporates auxiliary classifier generative adversarial networks (ACGAN), dynamic experience replay strategies, and Deep Q-learning Networks (DQN) to improve learning performance without additional data. Second, a task-oriented Meta-inverse reinforcement learning (Meta-IRL) approach is proposed to enhance adaptation to new tasks by leveraging encoders and exploring transformer-based multi-head and layer feature extraction strategies. Finally, a novel framework integrating a Global Attention Mechanism (GAM) with multi-layer feature fusion and latent Dirichlet allocation (LDA)

topic modeling is developed to enrich feature representations and optimize prompt generation in few-shot learning. Experimental validation on robot-mediated therapy tasks and other datasets demonstrates that the proposed methods enhance performance under limited data conditions. Overall, this work integrates efficient learning mechanisms with personalized intervention strategies, enabling the model to acquire richer and more informative representations that enhance its performance, while contributing to the broader goal of facilitating effective deployment of intelligent SARs in data-constrained, real-world environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABA	Applied behavior analysis
AC	Actor-critic
ACGAN	Auxiliary classifier generative adversarial networks
AI	Artificial intelligence
ASD	Autism spectrum disorder
BERT	Bidirectional encoder representations from transformers
BT	Behavior technician
CAS	Child's audio spectrogram
COS	Cosine similarity
DDPG	Deep deterministic policy gradient
DGAIL	Deterministic generative adversarial imitation learning
DNN	Deep neural network
DQN	Deep Q-Networks
DRL	Deep reinforcement learning
DTW	Dynamic time warping
ER	Experience replay
EX	Internal feature extractor
GAIL	Generative adversarial imitation learning
GAM	Global attention mechanism
GAN	Generative adversarial network
H-L	Multi heads and layers

HRI	Human-robot interaction
IN	Instance normalization
IRL	Inverse reinforcement learning
KL	Kullback-Leibler divergence
LDA	Latent dirichlet allocation
LfD	Learning from demonstration
LM	Language model
LMs	Language models
LLMs	Large language models
LM-RL	Language model and reinforcement learning
LN	Layer normalization
Meta-IRL	Meta-inverse reinforcement learning
MDP	Markov Decision Process
ML	Machine learning
MLM	Masked language modeling
NLP	Natural language processing
OOD	Out-of-distribution
PER	Prioritized experience replay
PLM	Pre-trained language models
PPO	Proximal policy optimization
RL	Reinforcement Learning
RLfD	Reinforcement learning from demonstration
RNN	Recurrent neural network

SARs	Socially assisted robots
SIA	Standard image augmentation
TD	Temporal difference
Word2Vec	Word to vector

CHAPTER ONE

INTRODUCTION

1.1 Learning from Demonstration in Socially Assistive Robots for Applied Behavior Analysis Therapy Scenarios

In recent years, advances in artificial intelligence (AI) and machine learning (ML) have significantly enhanced the capabilities of robotic systems. Among these developments, Learning from Demonstration (LfD) has emerged as a particularly impactful paradigm. LfD is a machine learning technique that enables robots to acquire skills by observing human demonstrations rather than relying on explicitly programmed instructions. Through various teaching modalities—such as kinesthetic guidance, teleoperation, and visual observation—robots can autonomously learn and replicate complex tasks demonstrated by humans. The versatility of LfD has led to its adoption in a broad range of domains, with social service robots becoming increasingly common in healthcare, customer service, entertainment, companionship, and education. A significant advantage of LfD is its capacity to facilitate the acquisition of tasks that are difficult to encode through conventional programming approaches. Tasks requiring fine motor control, dexterity, or precise adaptation across dynamic environments often exceed the capabilities of traditional rule-based programming. By enabling robots to learn directly from examples, LfD addresses these limitations and accelerates skill acquisition. Furthermore, it lowers the barrier for human-robot collaboration by allowing non-experts to teach robots new behaviors without extensive programming expertise [1].

Socially Assistive Robots (SARs) are specifically designed to engage with humans in social and interactive contexts. When integrated with LfD techniques, SARs can learn social behaviors and interaction patterns directly from human demonstrations, enabling them to dynamically adjust to real-time interactions, individual differences, and personal preferences. This adaptability results in improved learning efficiency, richer social engagement, and greater overall effectiveness compared to conventional programming methods.

One particularly promising application of LfD-enhanced SARs is in Applied Behavior Analysis (ABA) therapy for children with Autism Spectrum Disorder (ASD). Traditional ABA interventions often require lengthy one-on-one sessions with trained professionals. This approach is both resource-intensive and costly, and usually difficult to deliver in underserved or remote areas. SARs equipped with LfD capabilities can help address these challenges by enabling non-expert users to teach robots to provide personalized interventions without requiring specialized programming knowledge. Moreover, the capacity of LfD-enabled SARs to rapidly acquire new skills and adapt to changing therapeutic needs makes them particularly well-suited to addressing the diverse and evolving learning needs of individuals with ASD. Continued research into more efficient LfD-based approaches for SARs holds significant potential for expanding their use in personalized, accessible, and scalable therapeutic interventions [2], [3], [4].

In recent years, the Intelligent Robotics Laboratory has actively pursued research on various aspects of SARs, encompassing domain-specific technological requirements, systematic data acquisition protocols, expert-driven analytical methodologies, and

prospective strategies for future deployment. Among these efforts, the development of SAR systems based on LfD has emerged as a central focus in several collaborative projects.

To support investigations into practical application scenarios, one of the training datasets utilized in the laboratory originates from interactive sessions conducted during *wh*-question teaching interventions between behavior technicians (BTs) and children with ASD at a university-affiliated ABA clinic [5]. Figure 1. illustrates an example of the data collection process, depicting a robot-mediated learning scenario designed to teach *wh*-questions to a child with ASD. The corresponding examples of *wh*-questions are summarized in Table 1.

In this experimental setting, Behavior Technician 1 (BT1) operates pre-programmed control software in the background that directs the SAR to deliver appropriate speech and gestures during interaction with the child. It is important to emphasize that, within this configuration, the robot functions solely as a communication interface and does not exhibit autonomous learning or mobility capabilities. Simultaneously, Behavior Technician 2 (BT2) is tasked with supervising the interaction environment and managing potential emergencies to ensure safety and protocol adherence.

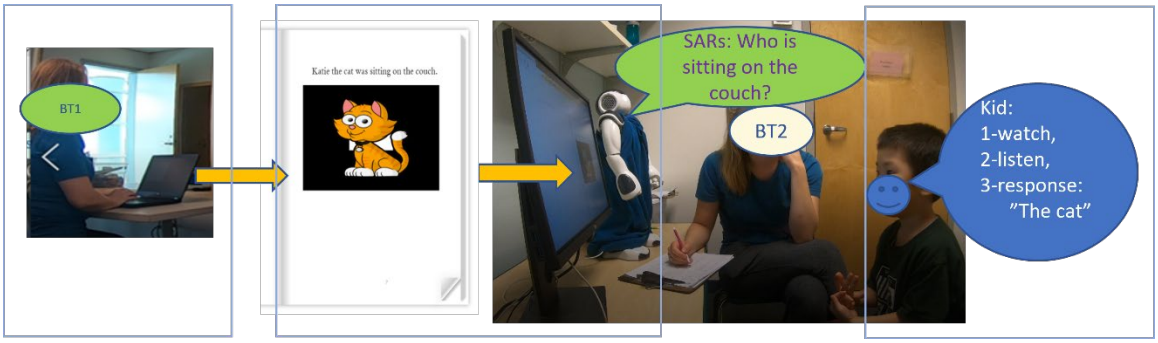
First, BT1 remotely presents a picture from a storybook to the child via a computer screen, as illustrated in Figure 1 (a). Subsequently, BT1 poses a *wh*-question through the SAR, for example, “Who is sitting on the couch?” After the question is delivered, BT1 observes and awaits the child’s response, as depicted in Figure 1 (b) and Figure 1 (c). During such interactions, a child’s responses generally fall into four main categories:

- Positive reinforcement through social praise for correct answers;
- Error correction for incorrect answers;
- Prompts for no response;
- Addressing other situations, such as the child being distracted or running away.

More specifically, if the child provides a correct response, positive reinforcement is delivered through verbal praise such as “Well done!” If the response is incorrect or absent, BT1 introduces a delayed prompt via the SAR to guide the child toward the correct answer. The objective of the session is for the child to correctly identify that the cat is sitting on the couch based on the visual stimulus presented on the screen. BT1 continues to pose the same *wh*-question and repeat the instructional process according to the child’s responses until the child successfully answers the question correctly in three consecutive trials.

This research starts from developing SAR models trained with LfD technology that can be deployed in similar therapeutic scenarios. By learning complex, abstract patterns of human interaction from real demonstrations, these SARs are designed to autonomously determine appropriate subsequent actions based on both current and historical interaction states—essentially replicating the decision-making process currently performed by BT1, without requiring backend human control or predefined programming. Such capability is crucial for enabling professionals without robotics expertise to use SARs to deliver individualized ABA interventions directly. The ultimate vision is for SARs to operate autonomously, providing adaptive therapeutic support without manual intervention. This work is driven by the challenges inherent in applying HRI learning to autism therapy

contexts. It is therefore centered on achieving efficient learning from a limited number of expert demonstrations in real-world HRI settings.



(a) (b) (c)

Figure 1. An example of an interaction scenario.

Table 1. Related wh-questions and interaction demonstrations.

Question type	**Sample**	**Prompts/Correct response**
Who	Who is sitting on the couch?	Katie/the cat
What	What is Katie the cat doing?	Sitting (on the couch)
Where	Where is the cat sitting?	(On the) couch
…	…	…

1.2 Problem Statement

Learning from Demonstration (LfD) methods have shown significant potential in enabling SARs to acquire new tasks and behaviors. However, their performance often declines sharply when demonstration data are scarce or suboptimal [6]. Acquiring a sufficiently large volume of demonstrations poses a significant challenge for real-world deployment, as data collection typically demands substantial time, human effort, and financial resources [3]. Moreover, access to demonstration data is challenging in sensitive

domains such as healthcare and education, where privacy regulations, ethical considerations, and security constraints further restrict data availability. For instance, in autism therapy scenarios, data collection is not only subject to stringent privacy protections but also requires complex medical review and approval processes. Consequently, in situations where extensive demonstration datasets are unavailable, it becomes essential to develop approaches that enable SARs to learn effectively and reliably from limited human demonstrations.

Unlike humans, who can leverage past experiences to learn quickly and generalize to new situations, SARs require substantial training data to achieve a comparable level of performance. Without large amounts of training data, SARs are unable to leverage prior knowledge as efficiently as humans do. Although SARs have achieved success on specific tasks, their generalization ability is limited. The current algorithms used to train SARs are designed to optimize their performance on specific tasks, and the resulting models lack the flexibility to adapt to new situations. This poses a significant challenge in real-world scenarios where the model needs to adapt to new tasks or situations. For instance, a SAR trained to support healthcare professionals by singing songs or reading stories may perform these tasks competently but will encounter difficulties when asked to engage in unfamiliar therapeutic activities or assist new patients. Such transitions typically demand substantial expert intervention, including new data collection, model retraining, or manual reprogramming. Therefore, enabling SARs to learn quickly and adapt to new tasks with minimal data is essential for their deployment in dynamic, real-world environments, where task requirements and user needs continuously evolve.

Given the limitation of collecting more demonstration data, on the one hand, improving data efficiency can be achieved by extracting more valuable data for training. On the other hand, by extending and enhancing the model and improving its learning capability, SARs models can acquire knowledge like humans do, enabling them to learn more complex, deeper, and abstract interactive rules with minimal demonstrations. This ability would allow the trained model to be applied to new tasks and environments and to perform well even on unfamiliar ones. This dramatically reduces the time and resources required to collect demonstration data, and, on the other hand, the improved learning capability of SARs makes it more feasible to apply them in the real world. Specifically, in the future, SARs could provide personalized therapy and support based on the diverse needs of ASD users or various courses.

1.3 Challenges

Because acquiring sufficient labeled demonstration data for model training in HRI-based ABA therapy scenarios is inherently tricky, and the dynamic, unpredictable nature of these interactions makes accurate simulation equally challenging. The objective of this study is to explore how to learn more from limited demonstration data. Specifically, "more" means being able to extract additional valuable information from the existing limited data using more effective methods, thereby enhancing the learning capacity of the SARs model. Namely, how to extract more features from limited data, which, when fed into the model, enables learning deeper or more abstract policies, thereby enhancing SARs' learning ability in low-data HRI scenarios. More specifically, the challenge primarily focuses on two aspects:

- **Network optimization:** Exploring methods to improve data efficiency and maximize the utilization of existing limited demonstration data involves investigating more efficient network structures based on limited data. Models often have limited learning capabilities due to scarce training data, leading them to learn only rules applicable to tasks similar to those in the training set. When both the training set and new tasks are limited, effectively leveraging existing large models and related techniques to update and enhance the current model is challenging for achieving better performance.
- **Maximizing information extraction:** The second challenge lies in maximizing information extraction from limited available data. This challenge requires innovative approaches to uncover crucial and latent information that might otherwise be overlooked. These techniques include exploring methods for extracting deeper features from the data, task-specific data augmentation, optimizing data utilization, and efficient integration, all aimed at improving the performance of few-shot learning.

1.4 Contributions

To learn more from Demonstration (LfD) with limited HRI scenarios, the main contribution of this study is to explore multiple methods from a modeling perspective based on existing limited training data, thereby improving data efficiency and further extracting useful information. A brief description of the contributions is listed below:

- Hierarchical Deep Q-Networks (DQN) are proposed, along with methods such as data augmentation and dynamic experience replay sizes to improve the sample efficiency.

- A task-oriented update method that combines meta-inverse reinforcement learning (Meta-IRL) and transformer encoder architectures is proposed. Through a systematic exploration of transformer encoder components—specifically the variation heads and layers configurations—feature extraction is optimized for sparse data representations.
- A novel framework incorporating a Global Attention Mechanism (GAM) that effectively integrates features from multiple layers of pre-trained language models, enhanced by Latent Dirichlet Allocation (LDA) generated topic features for prompt optimization.

1.5 Dissertation Outline

Chapter 2 presents the research relevant to the dissertation topic. It comprehensively reviews the literature related to improving data efficiency and extracting valuable information from limited demonstrations. Each section is introduced from both the network architecture and other methodological perspectives. Prompt optimization-related methods are also included.

Chapter 3 proposes a hierarchical Deep Reinforcement Learning (DRL) network with data augmentation techniques to improve sample efficiency when SARs are learning tasks from human demonstrations. The auxiliary classifier generative adversarial networks (ACGAN) are utilized to generate more demonstration data for training the models. The demonstration data size of the experience replay buffer is dynamically adjusted.

Chapter 4 introduces the Inverse reinforcement learning (IRL) approach to learn the expert policy, and a task-oriented weight updating method based on the meta-inverse reinforcement learning (Meta-IRL) structure to improve the performance of the new task adaptation. To maximize the value of limited training data, various transformer features with multiple heads and layers are investigated, providing practical guidance on selecting the optimal number.

Chapter 5 proposes a framework that incorporates a Global Attention Mechanism (GAM), leveraging multi-layer feature vectors from pre-trained language models to capture information that may have been overlooked during prompt generation, while simultaneously incorporating external latent features via topic keywords derived from third-party models.

Chapter 6 concludes by summarizing the various approaches proposed to enhance learning efficiency with limited data. It also offers suggestions for future research, highlighting the limitations of applying these methods in real-world settings and providing targeted insights from the experimental process.

CHAPTER TWO

LITERATURE REVIEW

The literature to be discussed primarily focuses on two aspects: improving data efficiency and extracting valuable information from limited demonstrations. Section 2.1 provides an overview of current approaches to improve the data efficiency of reinforcement learning from demonstration (RLfD), including network architecture and optimizing the storage and retrieval structures of samples. Section 2.2 primarily discusses methods for extracting valuable information from limited demonstrations, including approaches based on different network architectures, transformer-based methods, and techniques leveraging different layers of language models (LM) and supplementary information for LM. Finally, in Section 2.3, prompt optimization methods outperform traditional fine-tuning on low-resource datasets, offering compelling new opportunities to enhance HRI capabilities.

2.1 Improving Data Efficiency

2.1.1 Exploring Network Structures of Reinforcement Learning (RL)

Improving data efficiency in machine learning has been widely explored across fields such as computer vision, robotic manipulation, and learning from demonstration (LfD). A growing body of research shows that combining LfD with Deep Reinforcement Learning (DRL) significantly enhances sample efficiency, for example, in tasks like learning to play Atari games from demonstrations [7], [8], [9]. In these approaches, LfD provides the agent with practical prior knowledge and a more favorable initialization point, enabling

faster convergence and better initial performance than models trained solely through random exploration [10]. However, the effectiveness of LfD-based Deep reinforcement learning (DRL) still depends heavily on the quantity and quality of demonstration data [11]. Reinforcement Learning from Demonstration (RLfD) has therefore emerged as a key direction in this field, integrating demonstration data with reinforcement learning to improve sample efficiency and accelerate model learning [12].

A substantial body of research has focused on enhancing the data efficiency of Reinforcement Learning from Demonstration (RLfD) in recent years. For example, Active Reinforcement Learning with Demonstration (ARLD) introduces divergence- and variance-based query strategies derived from bootstrapped and noisy Deep Q-Networks (DQNs) to improve exploration efficiency and task performance [13]. Other studies have combined deep deterministic policy gradients with hindsight experience replay to bypass the inefficient random exploration phase in simulated robotic tasks [14]. The TD3 learning from a generator (TD3fG) approach enables agents to extract prior knowledge from demonstrations while mitigating issues related to poor Markovian properties in the data [10]. Additional efforts include sparse-constrained leveraged optimization methods based on proximal linearized minimization to handle demonstrations of mixed quality [15], and RL from demonstration via Bayesian network-based knowledge (RLBNK), which extracts structured knowledge from demonstrations to improve performance and data efficiency [14]. Deep Q-learning from Demonstrations (DQfD) pre-trains agents using a small amount of demonstration data before fine-tuning with a mix of demonstration and self-generated experience [7]. Similarly, Residual Reinforcement

Learning from Demonstrations (RRLfD) reduces exploration demands by integrating imitation learning with residual RL, enabling more efficient learning from sparse data [16].

Other research has explored data-efficient and interpretable LfD algorithms for learning social tasks [3], [17], [18], highlighting the potential of RL techniques for tasks such as reward function learning, task adaptation, and improved data utilization. Despite these advances, many of these methods are validated primarily in domains such as video games, robotic manipulation, and simulations. Even approaches that reduce reliance on pre-collected data still require substantial real-time interaction with the environment to achieve practical training. As a result, there is still no widely accepted solution to improve data efficiency in RLfD under extremely limited demonstrations, and significant challenges remain in maximizing the utility of such data in real-world applications [19].

On the other hand, Meta-Inverse Reinforcement Learning (Meta-IRL) infers reward structures directly from demonstrations. By leveraging a small number of therapist-provided examples, Meta-IRL can identify latent reward functions and transfer shared priors across tasks. To further enhance Meta-IRL's adaptability under data-scarce conditions, several optimization strategies have been proposed. For example, [20] models each task's reward as a distribution conditioned on a demonstrator-defined baseline, enabling inference of the most likely reward. However, sparse demonstrations and limited state–action coverage can lead to unstable estimates. Conservative reward-shaping methods like CRSfD [21] address out-of-distribution (OOD) issues, though they have primarily been evaluated in robotic manipulation. Other approaches, such as SMILe

[22] and probabilistic meta-policy learning [23], incorporate task-dependent probabilities or latent variables to improve policy learning from limited data. Despite these advances, many meta methods still rely on generic gradient-based meta-updates that overlook structural similarities across tasks. Meta-Adversarial Inverse Reinforcement Learning (Meta-AIRL) jointly optimizes reward functions and policies, achieving near-expert performance on unseen tasks [24]. Yet, when meta-testing tasks diverge significantly from meta-training ones, extensive fine-tuning is often needed [25], and negative transfer may occur if knowledge from source tasks is irrelevant.

2.1.2 Optimizing the storage and retrieval structures of training samples

Another widely explored strategy to improve the data efficiency of RLfD methods is to optimize the storage and retrieval mechanisms for training samples. Experience Replay (ER) and its extension, Prioritized Experience Replay (PER), improve sample efficiency and stabilize training in deep reinforcement learning by storing interaction experiences in a finite memory buffer and replaying them during training [26],[27], [28]. Much of the recent research has focused on understanding how the size and structure of the ER buffer [29] influence training performance and how replay mechanisms can help break temporal correlations to improve sample utilization [30]. However, buffer size remains a critical yet unresolved issue: excessively large or small buffers can significantly degrade algorithm performance [27], and increasing replay capacity or applying PER often has minimal impact when the agent already has a large memory [31]. To address these challenges, adaptive methods such as Adaptive Experience Replay (AER) [32], Combined Experience Replay (CER) [33], and Reservoir Sampling Double Replay

Buffer (RS-DRB) [34] have been proposed to dynamically evaluate the utility of stored experiences and automatically adjust buffer size. Other approaches, like Hindsight Experience Replay (HER) [30], focus on handling sparse and binary rewards by reinterpreting failed episodes as successful ones. At the same time, Forgetful Experience Replay (ForgER) [35] introduces a dynamic ratio of expert- and agent-generated samples to reduce reliance on high-quality demonstrations, as demonstrated in tasks such as Minecraft.

Despite these efforts, research conclusions regarding optimal ER size remain inconsistent across environments, and the underlying dynamics of experience replay are still not fully understood [36]. No universally accepted solution exists for determining the most effective ER size. Addressing this gap, one of the goals of this study is to explore the impact of ER size and develop an RLfD approach that leverages PER to optimize replay buffer configuration and improve sample efficiency.

2.2 Extracting valuable information from limited demonstrations

2.2.1 Network-based methods

To maximize the utility of limited demonstration data, several approaches have been proposed that target different aspects of the reinforcement learning pipeline, besides deeper exploration of the RL network, data augmentation, and optimizing sample storage and retrieval strategies [13], [37], [38], [39]. Some studies [40], [41] introduce additional networks to segment and filter valid demonstration data for more effective utilization. For example, Visual Generative Adversarial Imitation from Observation (VGAIfO-SO) [42] employs a state observer to infer additional state representations from high-dimensional

inputs, thereby enriching the available training data. Other research [43] explores soft expert guidance to improve RL performance when demonstrations are imperfect.

Despite their potential, these approaches typically require significantly more data than is available in Applied Behavior Analysis (ABA) therapy scenarios, which are characterized by small datasets and limited high-quality demonstrations. Attempts to enhance demonstration learning through contrastive learning and demonstration-label re-prediction [44] rely on pre-labeled positive and negative samples, which are impractical in small-data contexts involving classification and generation tasks like ABA therapy. Similarly, methods that improve efficiency by setting sub-goals during training [45] eliminate the need for expert data but rely heavily on the predefined structure of sub-goals. This dependency introduces substantial challenges when applied to ABA settings, where dialogue trajectories are complex and highly variable across individuals. The inherent variability in therapeutic interactions—where conversations aiming at the same teaching objective can differ dramatically—renders predefined sub-goal decomposition ineffective. Moreover, the intricacy of real-time HRI introduces further difficulties, as the model might pursue trajectories that fail to align with natural social interactions if constrained by rigid or coarse sub-goal definitions. As a result, these existing strategies fall short of effectively addressing the data scarcity and behavioral complexity inherent in ABA therapy environments.

2.2.2 Transformer-based methods

Other methods utilize transformers to extract more useful information. The researchers [46], [47] propose using Transformers for Meta-Reinforcement Learning

(TrMRL), leveraging the Transformer architecture to generate meaningful features. However, simply increasing the number of heads in the transformer is not always effective [48]. To address this issue, several studies [48], [49], [50], [51], [52], [53] have shown that pruning one or more attention heads or layers can preserve or even improve model accuracy, which was well-suited for large samples with consistent distributions. Further research [54] extends these findings, demonstrating that head pruning can improve performance, especially in scenarios with limited training data for the target language. Analysis of multi-head attention [51], [55], [56] reveals that different heads focus on distinct attention-related properties. While no single head excels across many relations, specific heads align closely with certain relations, often specializing in particular syntactic aspects [55]. In addition to focusing on the heads, the study [52] emphasizes the critical role of the lower layers in maintaining downstream performance. Anna Rogers [57] highlights that the lower layers of bidirectional encoder representations from transformers (BERT) retain the most information about the linear order of words. Research [58] shows that input tokens retain their identity in the initial hidden layers but gradually become less identifiable in the deeper layers. Meanwhile, the authors [59] show that multi-layer single-head attention can achieve effects similar to multi-head attention and often performs better, though at the cost of more complex training requirements. In summary, specific heads align closely with specific relations in multi-head attention, and lower layers contain rich, foundational information. Therefore, more research is needed on the strategic balancing of layers and heads in data-constrained scenarios.

2.2.3 Leveraging different layers of LM and supplementary information to LM

Recent research has also focused on extracting richer, more informative features from different layers of BERT [60] or by incorporating complementary information from external models. Manual parameter selection [61] and manually processed demonstration integration [62] are among the earliest approaches. Still, these methods often result in the loss of critical information due to the inherent limitations of manual feature design—especially in low-data scenarios. In contrast, the internal representations of large language models themselves can serve as a valuable source of information. Multiple studies [63], [64] have demonstrated that different layers within BERT capture distinct types of linguistic and semantic knowledge, with earlier layers often encoding crucial structural and fluency-related features that are not preserved in the final layer. Research also indicates that integrating features from multiple layers, rather than relying solely on the final layer, leads to significant performance gains [65].

Advanced techniques such as dynamic encoder fusion and knowledge distillation applied to decoder attention [66] effectively aggregate information from different BERT layers, enriching the model's representational capacity. These lower layers often retain sparse or abstract features essential for nuanced tasks, and selectively injecting prompts into specific layers—rather than across all layers—has been shown to improve performance [67]. Moreover, [68] emphasizes that targeting key layers for prompt optimization yields more efficient learning, particularly when training data is scarce. However, indiscriminate fusion of all layers can introduce redundancy and noise, especially when the dataset is small, where overlooked but valuable features from lower

layers may become diluted. This motivates the present study's exploration of selective multi-layer fusion strategies to enhance prompt optimization.

In parallel, alternative approaches have sought to enhance language models by integrating supplementary external models. For example, [69] employs a minimal set of aspect-specific seed words to guide model learning, combining them with BERT-based semantic similarity. Topic-informed BERT (tBERT) [70] incorporates topic modeling signals to improve semantic similarity prediction. At the same time, [71] proposes a BERT-LDA hybrid for more accurate topic extraction and sentiment analysis in online health communities. Similarly, [72] demonstrates improved text classification performance by combining weighted LDA with Word2Vec and BERT vectors. Building on these insights, this work investigates how supplementary models—particularly topic models such as LDA—can be effectively integrated into the LM-RL framework to boost prompt learning performance in low-resource environments.

2.3 Prompt optimization related methods

Most existing few-shot learning methods rely on a two-stage process: large-scale pre-training on diverse multitask datasets, followed by fine-tuning on smaller, domain-specific data. While this approach has proven effective, fine-tuning such large models remains technically demanding and requires substantial expertise [73], [74]. In response to these challenges, recent research has increasingly turned to prompts and instructions to guide large language models (LLMs), significantly improving their capacity to handle complex natural language processing (NLP) tasks. Prompt learning, which emphasizes adapting only a small set of task-specific parameters within pre-trained language models

(PLMs), has demonstrated notable success, frequently surpassing traditional fine-tuning methods in low-resource scenarios [74], [75]. This paradigm shift not only reduces computational and data requirements but also offers a more flexible and efficient pathway for enhancing human-robot interaction (HRI) systems, paving the way for more adaptive and context-aware robotic behaviors.

Recent state-of-the-art prompt-based methods have achieved strong performance through several strategies, including integrating labeled human feedback into prompt generation [76], [77], manual tuning of task-specific parameters [78], and techniques that leverage high-quality demonstrations or extract insights from ambiguous data [79]. Despite these advancements, adapting such methods to human-robot interaction (HRI) scenarios remains challenging. Designing effective prompts often requires substantial domain expertise and significant time investment [80]. Moreover, the inherently small size of HRI datasets limits their ability to represent the full diversity of possible contexts and tasks, leading to pre-training corpora that fail to capture the complexity of real-world interactions [81]. Consequently, prompt-based models frequently struggle to identify and utilize sparse yet critical information embedded within limited demonstration data—particularly when dealing with abstract, high-level, or ambiguous tasks—resulting in diminished performance and reduced effectiveness in downstream applications [82].

Most existing prompt optimization strategies aim to enhance model performance by either refining architectures or maximizing the utility of available data. Structural approaches such as P-Tuning v2 [68], Google's instruction tuning [83], and Prompt-DT [81] focus on adapting model architectures to improve prompt generation. Other

techniques, including APE [77], OPRO [84], and PromptBreeder [85], leverage powerful large language models (LLMs) such as PaLM2 [86] and the GPT series to propose and evaluate candidate prompts automatically. While methods like AutoPrompt [87] employ gradient-based search to iteratively refine prompts, they require direct access to model gradients, which is not always feasible. Moreover, many LLM-based approaches face significant limitations in interpretability, generalization, and cross-model reusability. Their heavy dependence on the size and diversity of fine-tuning datasets [83] further constrains their effectiveness in data-scarce contexts such as HRI, where demonstration data is inherently limited.

In contrast to conventional prompt-tuning methods, frameworks that integrate pre-trained language models with reinforcement learning (LM-RL) offer an alternative approach to prompt optimization that avoids computationally intensive gradient calculations. Recent applications [88] have explored this approach in areas such as aesthetic optimization for text-to-image generation, while Prompt-OIRL [89] applies offline inverse reinforcement learning to optimize query–prompt pairs. However, such methods often require a large volume of manually labeled rewards to train proxy reward models, which increases both resource demands and data dependency. Despite these challenges, LM-RL integration has shown considerable promise in improving prompt optimization efficiency. For instance, Prewrite [90] employs two LLM-based prompt rewriters trained via reinforcement learning to enhance downstream task performance. Similarly, RLPROMPT [91] and TEMPERA [92] leverage pre-trained language models within an RL framework, even with relatively small models, to achieve effective prompt

optimization. Notably, LM-RL architectures demonstrate strong performance independent of prompt pool size or the number of few-shot examples, making them especially well-suited for data-limited scenarios such as those encountered in human-robot interaction.

Beyond architectural advancements, several studies have explored integrating human feedback and expert demonstrations into prompt optimization, demonstrating performance improvements across various tasks [76], [77], [93]. However, these methods face notable constraints: they require extensive labeled feedback and demonstrations, entail significant computational and financial costs, and often perform poorly in small-data settings where bias effects are amplified.

To address these challenges, future research must go beyond optimizing LM-RL architectures and focus on maximizing the extraction of meaningful information from the limited data available. This involves developing innovative techniques to uncover hidden or latent features that traditional approaches might overlook. Effectively integrating this enriched information into the prompt-generation process holds significant potential to advance few-shot learning performance, particularly in data-scarce domains such as human-robot interaction.

CHAPTER THREE

IMPROVING SAMPLE EFFICIENCY WITH HIERARCHICAL REINFORCEMENT LEARNING NETWORKS

In real-world HRI and ABA therapy scenarios, particularly those involving children with Autism Spectrum Disorder (ASD), the first and most critical step of SAR-assisted interaction is recognizing the child's response. However, children with ASD often exhibit highly atypical speech patterns, including low volume, unstable prosody, irregular pauses, or non-standard semantic structures. Because of these unique characteristics, existing speech recognition models cannot be directly deployed for such therapeutic interactions, and large, population-specific datasets are rarely available due to privacy restrictions, high data-collection cost, and the difficulty of obtaining consistently labeled demonstrations from expert therapists. As a result, only a very limited number of demonstration samples can be collected to support model training. While existing data-hungry DRL exploration requires more demonstrations, which are not realistically obtainable in clinical ABA therapy settings, or large replay buffers introduce unnecessary computational overhead without substantial performance gains. In demonstration-scarce environments, the core bottleneck is the inefficient reuse of each sample during learning. Therefore, improving sample efficiency is essential for SARs to accurately recognize and classify children's responses.

Motivated by this application-driven necessity, this chapter proposes a data-efficient hierarchical DQN framework designed to "learn more from the same data." First, standard image augmentation (SIA) and Auxiliary Classifier Generative Adversarial

Networks (ACGAN)-based sample synthesis are applied to expand the dataset and mitigate overfitting, particularly for the ABA therapy dataset in which the original training samples—verbal responses from children with ASD—are extremely scarce. Second, a dynamic experience replay strategy is introduced to improve the utility of high-value transitions while reducing unnecessary computation compared to large, fixed replay buffers. Finally, a hierarchical DQN architecture performs layered decision refinement, enabling the model to resolve uncertain predictions and further improve recognition accuracy. In this way, the proposed method supports the first essential step in the target application: accurately identifying the child's response category during robot-mediated interaction. Experimental validation on two datasets—a real ABA therapy dataset containing atypical ASD speech and a secondary non-verbal emotion dataset—confirms that the proposed strategies significantly enhance model robustness and sample efficiency under limited demonstrations, thereby validating the effectiveness of the approach. Parts of this chapter are based on the author's previously published work [38]. The content has been adapted and extended for inclusion in this dissertation.

3.1 Sample Efficiency Improved Method via Hierarchical Reinforcement Learning Networks

A standard image augmentation (SIA) technique and an auxiliary classifier generative adversarial network (ACGAN) are employed to expand the dataset and maximize the utility of existing data, thereby enhancing model performance. As illustrated in Figure 2, following the optimization of experience replay (ER) size, a hierarchical reinforcement learning framework is introduced based on a Deep Q-Network (DQN) integrated with

prioritized experience replay (PER). This framework incorporates two DQNs: the first (DQN1) generates initial outputs, while the second (DQN2) performs targeted reprocessing of these outputs. This layered approach improves overall model performance without requiring additional demonstration samples.

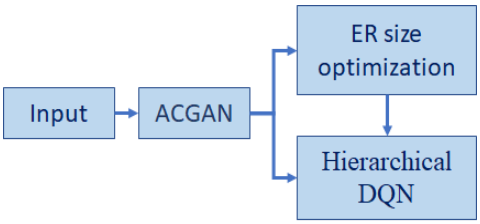


Figure 2. Methods modules.

3.1.1 Data augmentations

To address the challenge of learning effectively from limited demonstration data, standard image augmentation (SIA) techniques are first applied to expand and diversify the training set. These methods include flipping, rotation at multiple angles (45°, 90°, 135°, 180°), separation of RGB color channels, mirroring, darkening, brightening, blurring, and noise injection. In this approach, the original data is converted to image format for visualization, ensuring that essential features are retained and minimizing feature loss. Importantly, background noise is preserved rather than removed, as it can contain meaningful cues related to emotional states or behavioral patterns. For instance, subtle signals such as tapping on a desk might indicate nervousness or anger, and removing such noise could inadvertently eliminate valuable semantic, tonal, or affective information critical for accurate model learning.

Next, an Auxiliary Classifier Generative Adversarial Network (ACGAN) is applied to further expand the dataset by generating additional labeled samples for training.

ACGAN, a specialized variant of the standard generative adversarial network (GAN) [94], has proven highly effective in tasks such as image synthesis. In the ACGAN framework used within this LfD system, two networks—the generator (G) and discriminator (D)—are trained simultaneously. Each generated sample is associated with a class label $c \in C$ (representing the available classes) in addition to the noise vector z. The label c is encoded as a one-hot vector, serving as a conditional input to the generator. The generator then uses both the noise z and the class label c to produce synthetic images. The discriminator network receives the generated image along with its label and outputs two key pieces of information: the probability of the image being real or fake, and the predicted class label. This is achieved by incorporating a classification loss L_c alongside the standard GAN loss L_s, as defined in Equations (1) – (3). The combined loss enables the discriminator to assess authenticity and predict class categories simultaneously. For both real images (X_{real}) and generated images (X_{fake}), the discriminator thus learns not only to distinguish between real and synthetic data but also to classify them according to their labels. As a result, the system is able to generate a larger volume of demonstration data enriched with class information, significantly improving training efficiency and model robustness [3], [95].

$$L_s = E[logP(S = real|X_{real})] + E[logP(S = fake|X_{fake})] \quad (1)$$

$$L_c = E[logP(C = c|X_{real})] + E\left[\log \mathrm{P}\left(C = c\middle|X_{fake}\right)\right] \quad (2)$$

$$Loss = L_c + L_s \quad (3)$$

The combined use of SIA and ACGAN techniques expands the dataset by at least an order of magnitude, increasing the number of samples in each category by more than

tenfold. The precise degree of augmentation varies depending on the size and composition of the original dataset, and detailed results of this expansion are reported in the experimental section.

3.1.2 Deep Q-Networks and experience replay size optimization

After applying SIA and ACGAN to expand the training data, Deep Q-Networks (DQN) are employed as the foundational network to evaluate the effectiveness of data augmentation. DQN is a core deep reinforcement learning algorithm that integrates Q-learning with deep neural networks, enabling agents to handle high-dimensional and continuous state spaces where traditional tabular Q-learning becomes infeasible. It approximates the action-value function $Q(s, a; \theta)$ using a deep neural network parameterized by θ, allowing the model to estimate the expected cumulative reward for a given state-action pair. To improve training stability and convergence, DQN introduces two critical innovations:

- Experience Replay: Transitions (s, a, r, s') are stored in a replay buffer D, and mini-batches are randomly sampled during training. This breaks the temporal correlation between sequential samples, thereby improving data efficiency.
- Target Network: A separate network $Q(s, a; \theta^-)$ is used to compute the target value, reducing divergence during learning.

In this study, a Deep Q-Network (DQN) with experience replay (ER) is employed to enhance sample efficiency and stabilize training in deep reinforcement learning. ER achieves this by storing experiences in a finite memory buffer [27] and replaying them during training [28] [96]. Specifically, each demonstration is stored as an experience

tuple $e_t = (S_t, A_t, R_t, S_{t+1})$ in a replay buffer $D = \{ e_1, e_2, \ldots \}$, where S_t and S_{t+1} represent the current and next states, A_t is the action taken, and R_t is the corresponding reward. During the training process, mini-batches of experience samples are uniformly drawn from this buffer to update the network parameters, thereby improving learning efficiency and breaking the temporal correlations between consecutive samples. Beyond standard ER, this work also incorporates prioritized experience replay (PER), which further improves sample efficiency by prioritizing experiences based on their temporal difference (TD) error. The magnitude of a sample's TD error reflects how much the model can learn from it — larger errors indicate samples that are more informative and should be revisited more frequently. Rather than sampling uniformly [97], PER retrieves experiences from the replay buffer according to their priority, which is stored in a sum-tree data structure to enable efficient selection. As expressed in Equation (4), a small positive constant $\varepsilon > 0$ is added to ensure that every sample has a non-zero probability of being selected. By directing the training process toward transitions with higher learning potential, PER allows the agent to focus on samples that significantly impact policy updates. This prioritization accelerates convergence, improves learning stability, and enhances overall performance.

Figure 3 illustrates the architecture of the DQN framework enhanced with PER, showing how the prioritized sampling mechanism is integrated into the learning process.

$$Priority = P_i = \left| Q_{target} - Q_{value} \right| + \varepsilon \tag{4}$$

As discussed earlier, determining the optimal sampling size of the experience replay (ER) buffer is a crucial factor in shaping how stored samples are updated and utilized

during training. However, there is currently no widely accepted standard for selecting an appropriate buffer size. This issue becomes particularly significant in scenarios with limited demonstration data. If the ER buffer is maintained at a large, fixed size, training iterations can become considerably longer, yet this increased computational cost does not necessarily translate into substantial performance gains. To investigate this, the present work conducts a comprehensive evaluation of various ER buffer sizes within a DQN framework enhanced with prioritized experience replay (PER), examining how different capacities influence both training efficiency and model performance. Specifically, the study compares training iteration times and accuracy across a range of buffer sizes, both small and large. Based on these findings, the approach introduces a dynamic resizing strategy that adjusts the ER capacity throughout the training process rather than relying on a fixed size. This adaptive mechanism aims to achieve a balance between computational efficiency and performance improvement, ensuring more effective utilization of limited demonstration data.

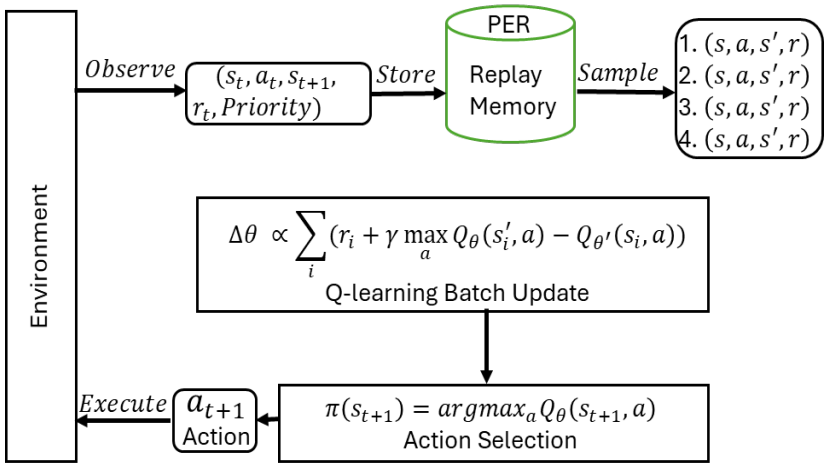


Figure 3. DQN with PER.

3.1.3 Hierarchical DQN

Building upon the DQN framework with prioritized experience replay (PER), a hierarchical DQN architecture is introduced. As illustrated in Figure 4, this design employs two distinct DQNs, each representing an independent policy that processes the same input from different perspectives. The concept is analogous to analyzing two similar images from separate viewpoints—such as left and right—where each perspective can yield unique insights.

For a given state input, DQN1 generates an output based on one perspective, while DQN2 produces an output from another. In some cases, the two networks may produce identical results, leading to consistent policy decisions. However, their outputs are also often different. In such situations, one network may offer a more effective action policy for specific inputs, whereas the other might perform better in various contexts. Because the strengths of each network vary depending on the input, directly merging their outputs or arbitrarily selecting one policy over the other is not an optimal solution. This observation motivates a hierarchical structure that intelligently leverages the complementary strengths of both networks rather than simply combining their outputs.

In this study, rather than directly selecting one output from the two networks, DQN1 and DQN2 are integrated into a hierarchical structure in which they operate in a sequential and interdependent manner rather than as isolated or parallel models. The combined input to these two DQNs can be represented as a tuple $M = (S_t, A_t, S_{t+1}, Q\_value_t, R_t, Priority_t)$, where S_t and S_{t+1} denote the current and next state spaces,

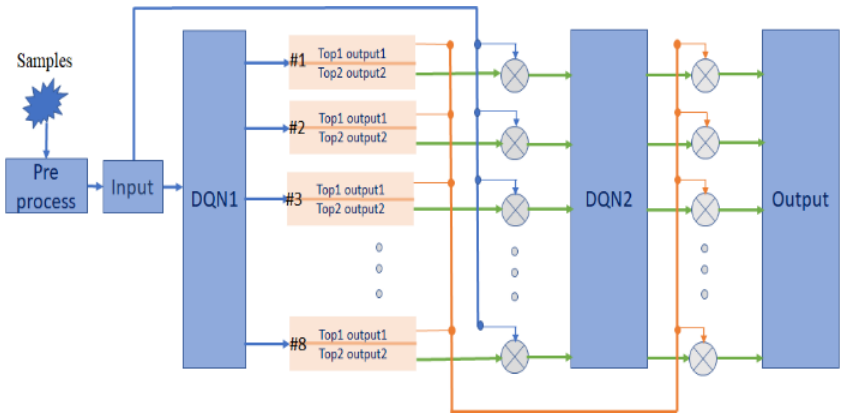


Figure 4. Network Structure.

respectively; A_t represents the action space; and $Priority_t$ corresponds to the priority weight (prio_weight) defined in Equation (4). Q_value_t is the action-value function Q (s, a) for a given state-action pair (s, a), while Q' represents the estimated Q_value_{t+1} for the next state-action pair (s', a'). The update rule for Q-values, shown in Equation (5), is influenced by three key factors: the current Q-value $Q(s, a)$, the reward $R(s, a)$, and the maximum predicted Q-value $Max_{a'}Q'(s', a')$ for the next state. The learning rate α controls the step size of the update, while the discount factor γ determines the importance of future rewards. The reward R is defined based on the specific task. For instance, when the input consists of images representing different emotional states, the objective of the DQN is to perform classification. In this case, the states correspond to the Learning from Demonstration (LfD) inputs, and the actions represent the classification labels (N). During training, if the action selected by the network matches the correct label, the reward is set to R = 1; otherwise, it is set to R = 0. This design enables the two DQNs to complement each other's decision-making processes, allowing the second network to refine and build upon the output of the first. By doing so, the hierarchical structure

leverages the strengths of both networks, resulting in improved decision quality and greater data efficiency.

$$newQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma Max_{a'}Q'(s',a') - Q(s,a)] \quad (5)$$

In the proposed hierarchical framework, DQN1 and DQN2 serve distinct yet complementary roles in improving decision-making and data efficiency. During the training phase of DQN1, the input consists of the entire state space S = { $S_1, S_2, S_3 \ldots, S_n$ } along with the corresponding action space A = { $A_1, A_2, A_3 \ldots, A_n$ }. After training, DQN1 learns a policy model based on the mapping ($S, A_1, A_2, A_3 \ldots, A_n$). DQN2 is trained more focusedly. Instead of using the entire action set, it only uses a subset of state-action pairs corresponding to the top two predicted actions from DQN1. Specifically, the input to DQN2 is S_{p+q} and the corresponding actions A_p and A_q, where p and q represent the indices of the top two Q-value predictions from DQN1. DQN2 then trains multiple specialized models for different state-action combinations based on varying reward signals and Q-values. For example, if there are four possible actions $\{A_1, A_2, A_3, A_4\}$, DQN1 generates a global model ($S, A_1, A_2, \ldots, A_4$). DQN2, in contrast, trains six separate models based on all top-2 action pairs: (S_{1+2}, A_1, A_2), (S_{1+3}, A_1, A_3), (S_{1+4}, A_1, A_4), (S_{2+3}, A_2, A_3), (S_{2+4}, A_2, A_4), (S_{3+4}, A_3, A_4).

The inference process involves two decision scenarios based on the Q-values produced by DQN1:

- Top-2 Q-values are close:

  If the two highest Q-values are nearly equal, it indicates that the model is uncertain between two possible actions (e.g., A_2 and A_4). In this case, DQN2 is employed to

perform a secondary evaluation to refine the decision. For the same input S_t, the model (S_{2+4}, A_2, A_4) — trained specifically for that action pair — is used to make the final selection. In other words, while DQN1 might predict either A_2 or A_4, DQN2 provides a more precise classification, such as confirming A_4 as the final action.

- Top-2 Q-values differ significantly:

  If there is a clear margin between the highest and second-highest Q-values, it suggests high model confidence in the top-1 prediction. In this scenario, the top-1 action from DQN1 is directly chosen as the output. In such cases, DQN1 and DQN2 consistently converge on the same decision, similar to recognizing two distinct objects from different perspectives.

Figure 5 illustrates the hierarchical DQN mechanism. DQN1 processes the complete state-action space, while DQN2 focuses exclusively on top-2 combinations for refined decision-making. Unlike traditional DQN approaches, which rely solely on the highest Q-value action, this hierarchical structure introduces an additional layer of targeted processing. The secondary evaluation performed by DQN2 acts as a form of focused, experience-based decision refinement, enhancing the model's ability to extract valuable information from limited demonstrations and significantly improving overall data efficiency.

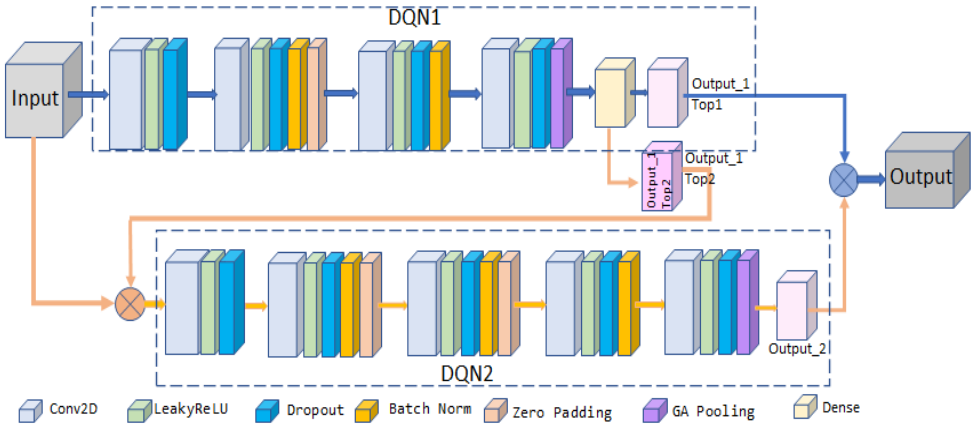


Figure 5. Hierarchical DQN.

3.2 Experiments

In the experimental evaluation, two distinct datasets were used to assess the effectiveness and robustness of the proposed approach under different data conditions. These datasets differ not only in size but also in the types and qualities of the audio data they contain.

- Dataset 1 (Verbal Audio): This dataset consists of verbal audio recordings collected directly from a real Applied Behavior Analysis (ABA) therapy environment. The recordings include natural background noise and have not undergone preprocessing such as noise filtering or normalization. As a result, Dataset 1 reflects a more realistic and complex training environment, capturing the variability and imperfections typically present in real-world HRI scenarios.
- Dataset 2 (Non-verbal Audio): This dataset contains non-verbal audio signals that are comparatively cleaner, less biased, and free from significant background noise. The data in this set are more controlled and consistent, providing a valuable contrast to the variability observed in Dataset 1.

By evaluating the approach on these two datasets, the experiments aim to investigate two key aspects: (1) the performance differences between verbal and non-verbal data when training reinforcement learning models for HRI tasks, and (2) the potential impact of data bias and background noise on learning outcomes. This dual-dataset evaluation provides valuable insights into the robustness and adaptability of the proposed method across varying data conditions.

The experiments are conducted in four phases:

- Data preprocessing in which raw files are transformed into spectrogram images of dataset 1 and dataset 2. Utilizing SIA and ACGAN to make data augmentation on the spectrogram images of dataset1, then make a comparison before and after the augmentation methods.
- Dynamic ER size optimization of dataset 1 set an appropriate ER size for the training procedure.
- DQN and other non-reinforcement learning networks (RNN and DNN) comparison, to choose which works the most effectively in the experiment.
- Hierarchical DQN on dataset 1 and dataset 2 compared to DQN.

Since Dataset 2 contains more samples than Dataset 1, the experiments involving Dataset 2 focused on two main aspects: data preprocessing and the implementation of the hierarchical DQN framework. In this case, additional data augmentation techniques such as SIA or ACGAN were deemed unnecessary. Instead, the emphasis was placed on evaluating how well the proposed hierarchical DQN approach performs when applied to a dataset that is already sufficiently large and relatively clean. Specifically, preprocessing

steps were applied to ensure data quality and consistency, including normalization, feature extraction, and formatting to suit the network input requirements. Following this, the hierarchical DQN model was employed to validate its effectiveness in leveraging available data to improve learning performance. This setup allowed the experiments to isolate and assess the contribution of the hierarchical DQN itself, without the confounding effects of augmentation, thereby providing a clearer understanding of its capabilities in scenarios with more abundant and less noisy data.

All experiments are performed on the deep learning framework TensorFlow, and utilize a computer with 16 GB of RAM, CPU Intel (R) Core (TM) i7-10750H 4.4 GHZ, and a 6 GB NVIDIA GeForce RTX 2060 GPU.

3.2.1 Datasets

3.2.1.1 Dataset 1: Emotion recognition intervention with children with ASD

The emotion recognition intervention dataset used in this study was initially collected in [3]. It captures therapist-guided emotion recognition sessions delivered to children with ASD through a teleoperated robot controlled via a virtual reality headset. In total, eight therapists conducted seventy-two demonstration trials with four children, illustrating real-world human-robot interaction in a therapeutic setting. Each trial consisted of three sequential steps: (1) the therapist, acting through the robot, presented an emotional expression solely using the robot's body language and asked the child to identify the robot's emotional state; (2) the child responded verbally to the question; and (3) the therapist, still teleoperating the robot, provided feedback or a follow-up response based on the child's answer.

From these demonstrations, 151 audio responses from the children were extracted for use in this research. Each response was manually annotated into one of eight emotional categories: “happy,” “none,” “scared,” “tired,” “other,” “angry,” “sad,” and “surprise.” This labeling provided the foundation for subsequent emotion classification experiments, enabling evaluation of the proposed methods on real-world therapeutic interaction data.

3.2.1.2 Dataset 2: The Variably Intense Vocalizations of Affect and Emotion Corpus (VIVAE):

VIVAE [98] is a comprehensive database of human vocal emotional expressions designed to capture a wide range of affective states across varying intensity levels. The dataset includes vocalizations representing three positive emotions—achievement or triumph, positive surprise, and sexual pleasure—and three negative emotions—anger, fear, and physical pain. All expressions were recorded under controlled experimental conditions to ensure high-quality audio and reliable emotional labeling.

In one experimental session of the VIVAE project, audio data from thirty participants were collected. For this study, which focuses on audio-based emotion analysis, a subset of 1,085 samples was used. Each sample is annotated with one of six emotion labels: “achievement,” “anger,” “fear,” “pain,” “pleasure,” or “surprise.” This dataset provides a diverse and well-balanced resource for evaluating emotion recognition models and testing their robustness across different affective states.

3.2.2 Data preprocessing and augmentation

As illustrated in Figure 6, the 151 audio responses collected from children in dataset 1 were converted into Child’s Audio Spectrogram (CAS) images, each with a resolution of

640×480 pixels. Every response was manually annotated with one of eight emotion categories: “happy” (19 samples), “none” (14 samples), “scared” (26 samples), “tired” (16 samples), “other” (19 samples), “angry” (21 samples), “sad” (22 samples), and “surprise” (14 samples). A subset of 24 samples was reserved as the test set. The remaining 127 samples were expanded using standard image augmentation (SIA) techniques such as blurring, brightening, darkening, flipping, noise addition, and rotations of 45°, 90°, 135°, and 180°. These augmentation steps increased the dataset by a factor of 10 for each class. Subsequently, an ACGAN model trained over 260,000 epochs was employed to synthesize an additional 50 samples per class.

Following preprocessing and augmentation, the final dataset contained 1,670 spectrogram images distributed across the eight emotion classes as follows: “happy” (210), “none” (160), “scared” (280), “tired” (180), “other” (210), “angry” (230), “sad” (240), and “surprise” (160). For dataset 2, which initially contained 1,085 audio samples, no augmentation was applied; instead, the raw audio files were directly transformed into spectrogram images for subsequent analysis.

Since DQN2 influences the overall performance of the model, its contribution was evaluated as part of the assessment of the complete Hierarchical DQN framework. The evaluation started with the baseline performance of DQN1 (as illustrated in Figure 5) and compared test accuracy before and after data augmentation. As shown in Figure 7, before

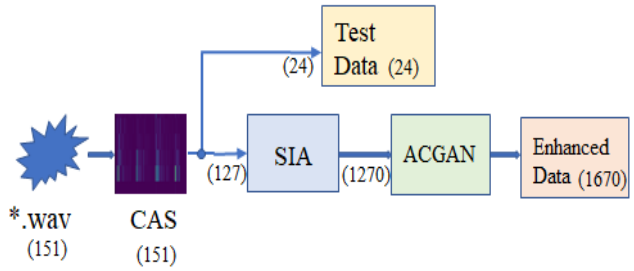


Figure 6. Dataset 1 preprocessing and augmentation.

augmentation on dataset 1, both training and testing accuracies initially increased; however, after approximately 150 epochs, the model exhibited clear overfitting. At this point, the training accuracy approached 0.9, while the testing accuracy plateaued around 0.3. In contrast, after applying data augmentation and extending training to 1,000 epochs, the degree of overfitting was substantially reduced. The testing accuracy curve began to closely follow the training accuracy curve, with the training accuracy stabilizing near 0.7 and the testing accuracy exceeding 0.6. This demonstrates that data augmentation effectively mitigates overfitting and significantly enhances the model's generalization performance.

3.2.3 Dynamic ER size optimization

To evaluate the effect of experience replay (ER) buffer size on model accuracy and identify the optimal configuration for Dataset 1, ER sizes ranging from 1 to 128 were tested using DQN1. For clarity, Figure *8* illustrates the test accuracy results for representative buffer sizes of 16, 32, 64, and 128, with all accuracy values measured after 1,000 training epochs. The results show that an ER size of 16 yields the lowest training and testing accuracy, indicating that a small buffer limits the diversity of replayed experiences and negatively affects learning performance. In contrast, ER sizes of 32, 64,

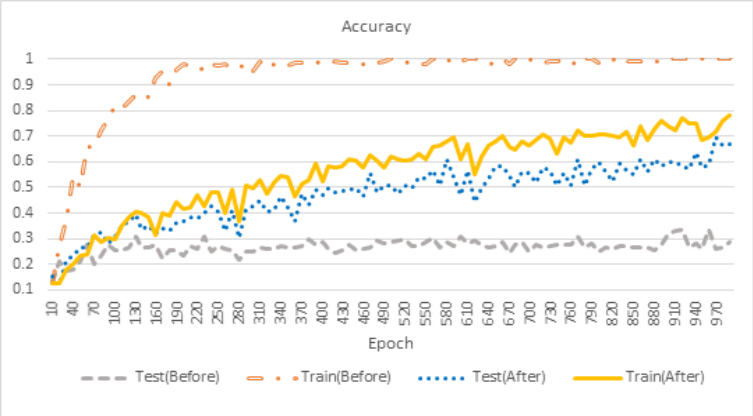


Figure 7. Accuracy before & after augmentation on DQN.

128 produces significantly higher accuracy, and after approximately 500 epochs, their test performance becomes nearly identical. Table 2 presents the average time per training iteration (T: s/it), revealing that larger ER sizes increase computational overhead.

To balance performance and efficiency, a dynamic ER strategy with buffer sizes ranging from 4 to 32 was implemented. As shown in Figure 9, after around 500 epochs, the test accuracy achieved with the dynamic ER approach is comparable to that of the fixed-size ER buffer at 128. However, the dynamic approach reduces training time substantially, with total training time dropping to less than two-thirds of that for a fixed ER size of 32 ($t_{32} = 23min55s$, $t_{dynamic} = 15min15s$, epoch = 600). This demonstrates that a dynamic ER configuration effectively maintains model performance while significantly improving computational efficiency.

Table 2. Dynamic ER size.

ER size	4	8	16	32
T(s/it)	1.05	1.35	1.8	2.43
Epoch	1-99	100-299	300-399	400-1000

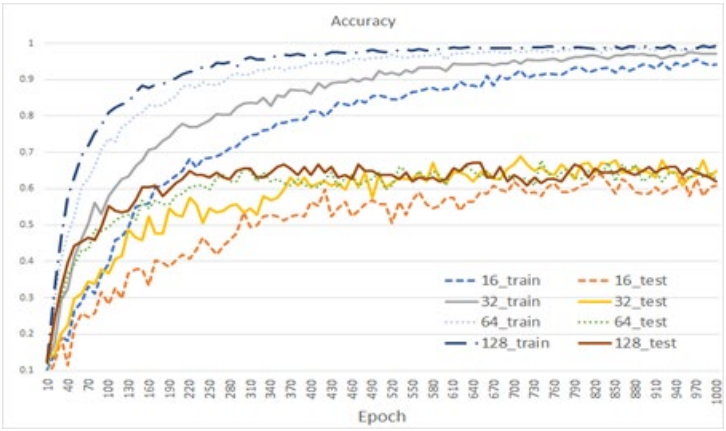


Figure 8. Accuracy scores of different ER sizes.

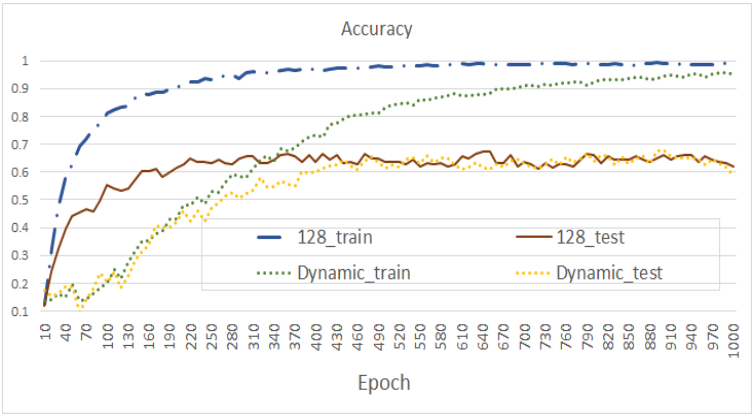


Figure 9. Accuracy scores of ER size 128 and dynamic ER sizes.

3.2.4 CNN, RNN vs DQN1

A comparative evaluation was performed among three models—DNN, RNN, and the proposed DQN1 (illustrated in Figure 5)—to assess their relative performance under the

same experimental conditions. As shown in Figure 10, the DNN architecture follows the design described in [3], consisting of two convolutional layers, dense layers with ReLU activation, and three LSTM layers. The RNN model is constructed using a GRU layer, followed by a 32-unit dense layer with ReLU activation, a 64-unit dense layer, and a final softmax output layer.

The performance comparison results are presented in Figure 11. Due to the limited size of the dataset, both DNN and RNN models exhibit clear signs of overfitting after approximately 200 epochs, with their testing accuracy plateauing around 0.5. In contrast, the DQN1 model shows a significant reduction in overfitting and demonstrates more stable training behavior. After 1,000 epochs, DQN1 achieves training accuracy exceeding 0.7 and testing accuracy surpassing 0.6.

Overall, the results clearly demonstrate that DQN1 outperforms both DNN and RNN models in terms of generalization and predictive accuracy, highlighting its effectiveness in handling limited data scenarios and improving performance in reinforcement learning-based tasks.

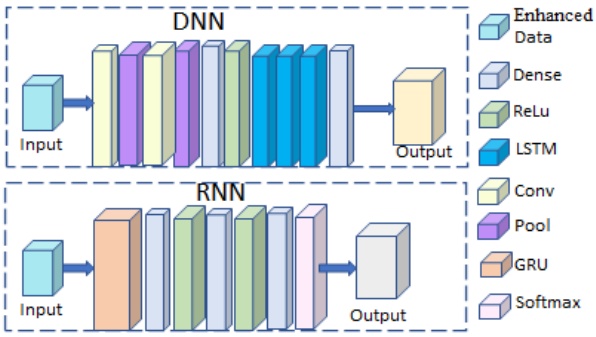


Figure 10. DNN & RNN network structure.

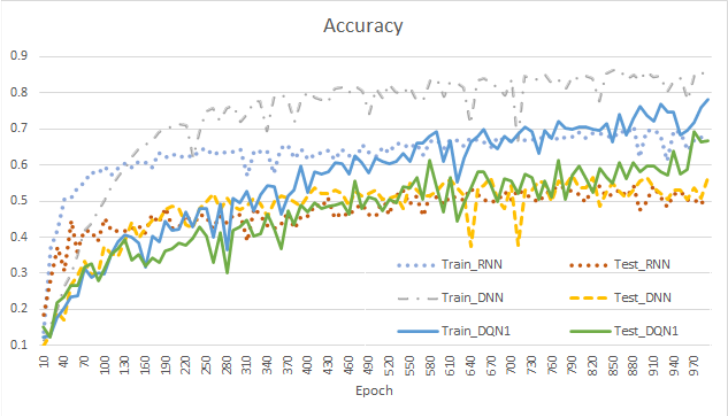


Figure 11. DNN, RNN, and DQN1.

3.2.5 Hierarchical DQN

3.2.5.1 DQN1 and DQN2

In the experimental setup, the state space S corresponds to the input spectrogram images derived from the datasets. For dataset 1, the inputs to DQN1 consist of the enhanced set of 1,670 image samples distributed across 8 emotion categories. For dataset 2, the input consists of 1,085 image samples representing 6 emotion categories. The action space A is defined by the class labels associated with each dataset. Specifically, for dataset 1, the action set is A = {“happy”, “none”, “scared”, “tired”, “other”, “angry”, “sad”, “surprise”}, while for dataset 2, the action set is A = {“achievement”, “anger”, “fear”, “pain”, “pleasure”, “surprise”}. The reward function R follows a sparse binary scheme. A positive reward of +1 is assigned when the chosen action correctly matches the target class label. Conversely, if the action corresponds to an incorrect class, the episode terminates immediately, and the reward is set to 0. This sparse reward structure emphasizes accurate classification and encourages the agent to learn effective decision-making policies.

As illustrated in Figure 5, the hierarchical DQN framework comprises two components: DQN1 and DQN2. To evaluate the effectiveness of the hierarchical structure, a comparative analysis was performed between the baseline network (DQN1 only) and the enhanced configuration (DQN1 + DQN2). DQN1 is built using a convolutional neural network (CNN) architecture with four convolutional layers (kernel size: 3×3) configured with 128, 64, 64, and 16 filters, respectively. The network includes two Batch Normalization layers (momentum = 0.8) to stabilize and accelerate training, four Dropout layers (0.25) to mitigate overfitting, and LeakyReLU activation functions to improve gradient flow. ZeroPadding layers are incorporated to preserve spatial dimensions, followed by a GlobalAveragePooling layer for dimensionality reduction before output. DQN2 adopts a deeper CNN architecture, consisting of five convolutional layers (kernel size: 3×3) with 128, 64, 64, 64, and 32 filters, respectively. It includes three Batch Normalization layers (momentum = 0.8), five Dropout layers (0.25), and standard ReLU activation functions. Similar to DQN1, ZeroPadding and GlobalAveragePooling layers are applied to maintain spatial integrity and aggregate feature maps. This hierarchical design enables DQN2 to refine secondary decisions based on the top candidate outputs generated by DQN1, thereby improving classification precision and overall model performance.

As outlined in the proposed methodology, the hierarchical decision-making process involves two sequential stages. First, DQN1 performs the initial recognition step by processing the input sample and generating the top2 prediction results based on the highest Q-values. In the second stage, DQN2 conducts a focused refinement on these two

candidate classes to improve classification accuracy. For example, in dataset 1, an input image is first processed by DQN1, which outputs the two most likely categories—such as "tired" and "other"—corresponding to the highest Q-values. This same image is then passed to DQN2, which performs enhanced recognition specifically between these two predicted categories. The final output is determined by combining the results of DQN1's broad initial classification and DQN2's targeted refinement, enabling more accurate decision-making than either network could achieve independently.

3.2.5.2 Evaluation Metrics and Results

To assess the effectiveness of the proposed hierarchical DQN, two key evaluation metrics are employed: testing accuracy and F1 score [4], [99]. These metrics are used to compare the performance of the baseline DQN model against the hierarchical DQN configuration, serving as the conditions for the dependent variables. Furthermore, a bootstrap paired t-test is performed to statistically validate the performance differences, testing the following hypotheses:

- H1: Accuracy is higher using hierarchical DQN than using DQN.
- H2: F1 score is higher using hierarchical DQN than using DQN.

For dataset 1, after 1000 training epochs, accuracy and F1 scores were obtained from 34 saved models (recorded at 30-epoch intervals from epoch 1 to 1000). The results are illustrated in Figure 12 and Figure 13, which compare the performance of the baseline DQN with the proposed hierarchical DQN approach. As shown in Table 3, the hierarchical DQN consistently outperforms the standard DQN in both evaluation metrics. Statistical analysis further confirms these findings, revealing significant differences in

both accuracy (ρ = 0.001) and F1 score (ρ = 0.01). These results strongly support hypotheses H1 and H2, demonstrating that the hierarchical DQN structure significantly enhances model performance in limited-data scenarios.

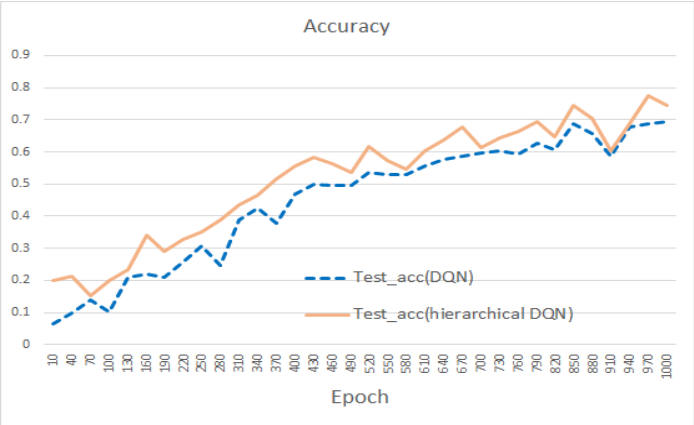


Figure 12. Accuracy of DQN and Hierarchical DQN (Dataset 1).

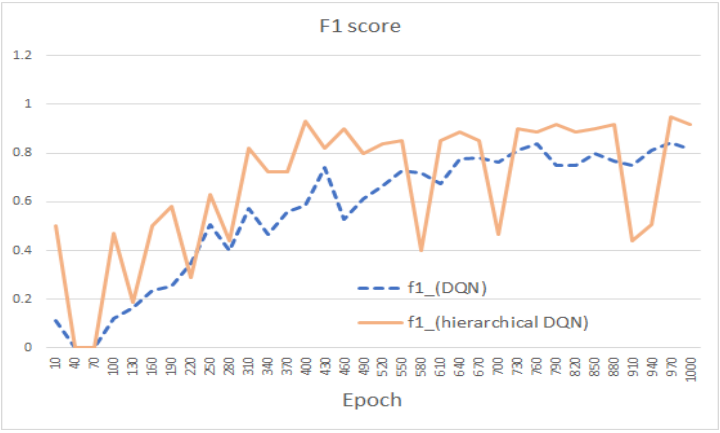


Figure 13. F1 scores of DQN and Hierarchical DQN (Dataset 1).

For dataset 2, after 1200 training epochs, accuracy and F1 score metrics were collected from 30 saved models (recorded every 40 epochs from epoch 1 to 1200). The performance comparison is illustrated in Figure 14 and Figure 15. As summarized inTable 4, the hierarchical DQN consistently achieves higher average accuracy compared to the standard DQN. Statistical analysis further confirms the superiority of the hierarchical approach, revealing significant improvements in both accuracy ($\rho = 0.001$) and F1 score ($\rho = 0.001$). These results provide strong empirical support for hypotheses H1 and H2, demonstrating the effectiveness of the hierarchical DQN in improving model performance on dataset 2.

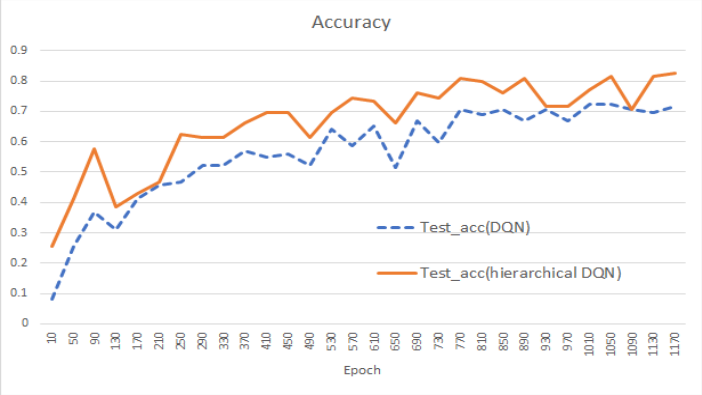


Figure 14. Accuracy before1 and after2 Hierarchical DQN (Dataset 2).

Table 3. Bootstrap paired t-test bca 95% results (dataset 1).

		DQN	Hierarchical DQN
Accuracy	*Mean*	0.451	0.515
	SE	0.033	0.031
	t(33)	-10.203	
	Sig.	**0.001**	
F1 score	*Mean*	0.566	0.667
	SE	0.045	0.048
	t(33)	-3.15	
	Sig.	**0.01**	

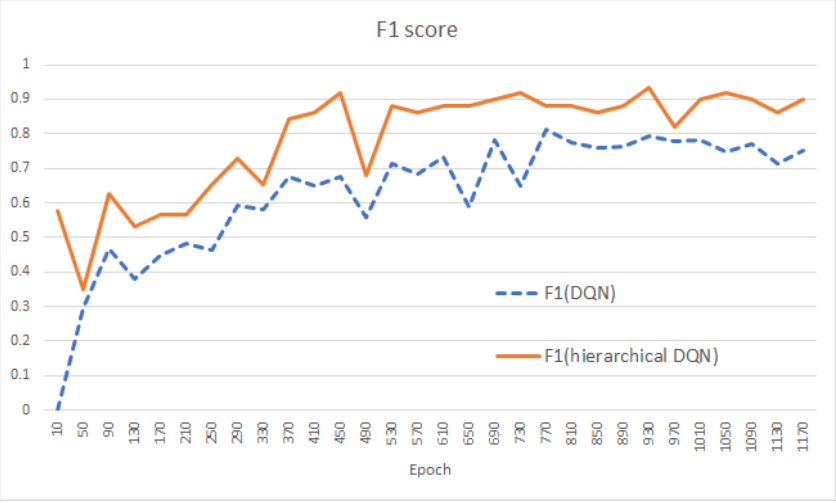


Figure 15. F1 scores before1 and after2 Hierarchical DQN (Dataset 2).

Table 4. Bootstrap paired t-test bca 95% results (dataset 2).

		DQN	Hierarchical DQN
Accuracy	Mean	0.566	0.665
	SE	0.029	0.026
	t(29)		-10.166
	Sig.		**0.001**
F1 score	Mean	0.629	0.787
	SE	0.033	0.028
	t(29)		-8.851
	Sig.		**0.001**

3.3 Results and Discussion

For dataset 1, applying the SIA and ACGAN methods significantly mitigated overfitting and boosted testing performance, with accuracy rising from 0.28 to 0.51 at 500 epochs and from 0.29 to 0.66 at 1000 epochs. The adoption of dynamic ER sizes further enhanced efficiency by balancing performance and training time, reducing iteration time from $t_{32} = 23min55s$ to $t_{dynamic} = 15min15s$. The introduction of the hierarchical DQN brought additional improvements, with testing accuracy increasing from 0.53 to 0.61 at 520 epochs, from 0.61 to 0.64 at 820 epochs, and from 0.69 to 0.74 at 1000 epochs. Statistical analysis using a t-test confirmed that both accuracy and F1 score improvements after implementing the hierarchical framework were statistically significant, demonstrating the effectiveness of the proposed approach.

For dataset 2, the comparison between the standard DQN and the hierarchical DQN shows a clear performance improvement: testing accuracy increased from 0.53 to 0.69 at 530 epochs, from 0.68 to 0.79 at 810 epochs, and from 0.71 to 0.82 at 1200 epochs. Statistical analysis using the t-test confirmed that these improvements in both accuracy and F1 score were significant.

The experimental results indicate that the proposed methods consistently enhanced performance across different training durations. However, the gains in accuracy and F1 score for dataset 1 were smaller than those observed for dataset 2, and the F1 score fluctuations were also more pronounced in dataset 1. This discrepancy is likely due to the inherent differences in data quality: dataset 1 was collected from a real ABA therapy environment without preprocessing, meaning that background noise and other environmental factors were retained, while dataset 2 consisted of cleaner, less biased audio samples. In real-world applications, such data sparsity and noise are inevitable. Thus, although the hierarchical network improved performance on both datasets, external factors—such as background noise and bias—must be addressed to enhance model robustness and generalization further.

In summary, the proposed framework enables SARs to learn robust classification policies from extremely limited ABA demonstration data. By expanding scarce ASD speech samples via SIA and ACGAN, improving sample reuse with dynamic experience replay, and refining uncertain decisions via hierarchical DQN, the model achieves higher accuracy, reduced overfitting, and stronger generalization than conventional DQN-based LfD methods. Solving this recognition problem marks the first essential step in the target

ABA application, laying the foundation for subsequent task adaptation and interaction strategies explored in the following chapter.

CHAPTER FOUR

A TASK-ORIENTED WEIGHT UPDATE METHOD WITH ANALYSIS OF MULTI-HEAD AND LAYER FEATURE COMBINATIONS

After recognizing the child's response in Chapter 3, the next essential step in the ABA therapy pipeline is for the model to determine how a behavior technician (BT) should respond in order to guide the child toward the teaching objective. In real ABA sessions, BTs adapt their next utterance based on both the current response and the historical interaction trajectory, resulting in diverse and task-specific dialogue flows. However, each child's data typically contains only a limited number of dialogue instances, and the substantial behavioral variability across children makes it difficult to rely on large-scale, generic language corpora commonly used in traditional Natural language processing (NLP) tasks. Therefore, the model must be able to rapidly adapt to different interaction tasks despite having very few demonstrations.

While large language models (LLMs) can exhibit strong generative capabilities with few-shot examples [74], their dependence on pre-trained data distributions constrains both their adaptability to specific tasks and their interpretability in ABA therapy. In contrast, Meta-Inverse Reinforcement Learning (Meta-IRL) provides a more suitable paradigm by inferring task-dependent reward structures directly from limited therapist demonstrations, enabling knowledge transfer across related tasks. Although transformer encoders naturally contain rich multi-level representations, existing approaches typically rely only on final-layer embeddings, overlooking valuable structural and semantic cues distributed across different heads and layers. This results in suboptimal task adaptation,

particularly in low-resource HRI scenarios. Motivated by this observation, this chapter proposes a Meta-IRL framework that selectively leverages multi-head and multi-layer transformer features to improve task adaptation and generate the following BT response. By emphasizing the most informative internal representations and learning from prior interaction trajectories, the proposed method enables the model to behave more like a real BT—responding appropriately to the child and continuing the therapy process even with scarce demonstrations.

Experimental evaluations further demonstrate that the proposed method significantly improves the performance of RL models trained with a limited number of expert demonstrations collected from ABA therapy scenarios. These experiments are not conducted to verify whether the model can operate under the same constraints faced in scarce child-specific data, high variability in interaction patterns, and the need for task adaptation. Parts of this chapter are based on the author's previously published work [100]. The content has been adapted and extended for inclusion in this dissertation.

4.1 A Task-Oriented Weight Update Method with Analysis of Multi-Head and Layer Feature Combinations

In this work, the proposed Inverse Reinforcement Learning (IRL) framework builds upon the integration of Deep Deterministic Policy Gradient (DDPG) [101] with deterministic Generative Adversarial Imitation Learning (DGAIL) [102]. To improve the system's ability to adapt to new tasks, a task-oriented weight updating strategy is introduced in the outer loop of the Meta-IRL structure. In addition, a transformer-based encoder [66] is incorporated to enhance feature representation for RL training, as its

internal multi-head and layer mechanisms enable the model to capture more informative, task-relevant patterns from sparse data.

4.1.1 Deep deterministic policy gradients and generative adversarial imitation learning

The Deep Deterministic Policy Gradient (DDPG) algorithm, an off-policy and model-free approach, is particularly suitable for interactive settings where the distribution of input states cannot be predetermined. By leveraging deterministic policy gradients, DDPG accelerates the training of Generative Adversarial Imitation Learning (GAIL), which focuses on deriving the reward function from demonstration data shaped by policy behavior. Within the proposed framework, DDPG functions as the policy generator, producing action strategies that drive the internal Meta-RL network. It employs an actor–critic (AC) architecture to estimate value functions and update network parameters. Both the actor and critic components include a main network and a target network, with the latter (μ', Q') updated according to Equations (6) and (7), while the main network (μ, Q) is optimized using Equations (8) ~ (11). To enhance sample efficiency and stabilize training, state–action pairs generated by DDPG (τ_E) together with expert demonstration trajectories (τ_D) are stored in an Experience Replay (ER) buffer [27].

The actor component of DDPG is responsible for learning the action policy μ and updating its parameters θ using the policy gradient defined in Equation (8). The first term of this objective represents the action loss, where the updated policy is derived from the actor's main network (μ) based on the current state s_i. Here, $\bar{\mu}_{\tau E}$ denotes the mean action across the state–action pairs in τ_E. After applying gradient updates, the actor parameters θ^{μ} are iteratively adjusted toward their optimal values.

The second component of the equation (8) corresponds to the action-value loss, which is estimated by the critic's main network (Q). This network evaluates the value of the actions chosen for the current state s_i, while the actor network selects the optimal action according to $a_i = \mu(s_i; \theta^\mu) + N_i$, where N_i represents Gaussian noise added for exploration purposes. This interaction helps guide the actor toward producing more optimal policy outputs.

Finally, the third component captures the influence of the GAIL discriminator $D(s, a; \theta^D)$ on the actor's learning process. The hyperparameters $\{\delta_A, \delta_C, \delta_D\}$ are introduced to balance the contributions of the three terms, ensuring stable and effective policy optimization.

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{6}$$

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{7}$$

$$Loss_{actor_i}(\mu|s_i) = \delta_A \hat{\mathbb{E}}_{\tau E}(\mu(s_i;\theta^\mu) - \bar{\mu}_{\tau E})^2 - \delta_C \hat{\mathbb{E}}_{\tau E}\big(Q(s_i, a_i; \theta^Q)\big) - \delta_D \hat{\mathbb{E}}\big(log(\gamma_i)\big) \tag{8}$$

$$\gamma_i = D(s_i, \mu(s_i;\theta^\mu); \theta^D) \tag{9}$$

As shown in Equation (10), the value y_i is derived from the critic's target network $(Q'(s, a; \theta^{Q'}))$, which evaluates the expected return by incorporating the current discount factor γ and the next state (s_{i+1}), predicted by the actor's target policy $\mu'(s_{i+1}; \theta^{Q'})$. Beyond the critic's target network, the term γ_i in Equation (9) also plays a significant role in directing the learning dynamics of DDPG, as it is shaped by the feedback from the GAIL discriminator (D). To train the critic network (Q) effectively, its parameters are optimized by minimizing the loss function defined in Equation (11).

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'\big(s_{i+1}; \theta^{\mu'}\big); \theta^{Q'}) \tag{10}$$

$$\text{Loss}_{critic\_i}(Q|s_i, a_i) = \widehat{\mathbb{E}}_{\tau_{E,}\,\tau_D}\,(y_i - Q(s_i, a_i; \theta^Q))^2 \tag{11}$$

$$\widehat{\mathbb{E}}_{\tau_E}(\log\,(D(s_i, a_i))) + \widehat{\mathbb{E}}_{\tau_D}(\log\,(1 - D(s_i, a_i))) \tag{12}$$

As the discriminator, GAIL leverages a learned reward function to guide the DDPG generator toward aligning its behavior with the expert demonstration policy. The discriminator is trained by minimizing the loss function defined in Equation (12). To accelerate the training process, improve stability, and enhance adaptability to unseen tasks, this work integrates DDPG into the GAIL framework and employs the combined structure as the internal Meta-IRL network. The overall architecture of this integrated design is illustrated in Figure 16.

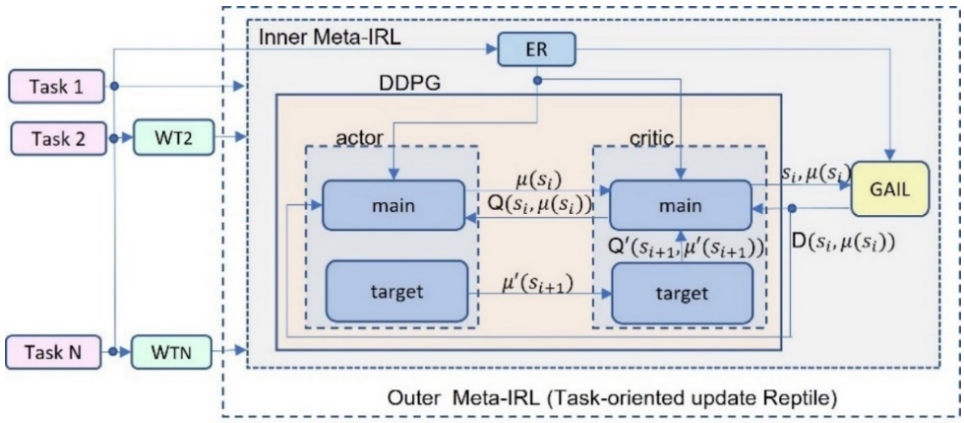


Figure 16. Network structure.

*Inner Meta-IRL: DDPG serves as a generator, GAIL learns the reward function. Outer Meta-IRL: Task-oriented update Reptile.

4.1.2 Task-oriented update method for Meta-IRL framework

Meta-learning is an effective paradigm for enabling models to rapidly adapt to new environments by exploiting structural similarities across multiple tasks. In this study, Reptile [103] is adopted as the core meta-learning framework, which learns an optimized initialization for model parameters through multiple gradient-based updates while

avoiding the computational complexity associated with second-order derivatives. This approach allows the meta-learner to apply distinct update procedures within the inner and outer loops of the training process, improving generalization to unseen tasks.

Specifically, the inner loop is responsible for optimizing model parameters on each individual task T_i, while the outer loop focuses on refining the initialization across a distribution of tasks T: $\{T_1, T_2, \ldots, T_n\}$. The batch version of the Reptile algorithm is defined in Equation (13), where Φ denotes the initial model parameters. After performing k steps of gradient descent or Adam optimization on data from a specific task T_i, the updated parameter state is represented as $\widetilde{\Phi}_i = U_{T^i}^k(\Phi)$. Here, ϵ is a step-size hyperparameter that controls the magnitude of the meta-update. This formulation enables the meta-model to accumulate useful inductive biases across multiple tasks, thereby improving adaptability to new and unseen environments.

$$\emptyset \leftarrow \emptyset + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\widetilde{\Phi}_i - \Phi) \tag{13}$$

Since conventional approaches such as gradient averaging or the addition of context distribution elements [22], [23], [104] are not well-suited to HRI scenarios with limited training data, a task-aware weight updating mechanism is introduced to incorporate the correlation between tasks into the meta-learning process. As described in Equations (14) ~ (15), the weight W_{Ti} is computed using three complementary measures of task similarity. Kullback–Leibler divergence (KL) quantifies the divergence between two probability distributions [105], Cosine similarity (COS) captures the degree of semantic similarity between document topics [106], and Dynamic Time Warping (DTW) measures the alignment between two sequences of unequal length [107].

In Equation (15), $D_{KL(Ti,\ Tj)}$, $D_{COS(Ti,\ Tj)}$, and $D_{DTW\ (Ti,\ Tj)}$ denote the KL, COS, and DTW similarity values, respectively, between tasks T_i and T_j. Equation (16) defines k as the set of non-repeating elements shared by both tasks. The term $(q_u,\ c_v)^2$ represents the Euclidean distance between q_u in T_i : $[q_1,\ q_2,\ \ldots,\ q_u,\ \ldots,\ q_n]$ and c_v in T_j : $[c_1,\ c_2,\ \ldots,\ c_v,\ \ldots,\ c_m]$, where q_u and c_v correspond to individual elements within the task sequences. The parameters $\{\alpha,\ \beta,\ \gamma\}$ are applied to normalize these three similarity measures from distinct perspectives.

To mitigate the limitations posed by small training datasets, additional random coefficients are introduced as scaling factors for these parameters, balancing their relative influence and reducing potential bias. After each internal training iteration for a given task, the outer-loop parameter update rule is formulated as shown in Equation (17). This strategy significantly decreases the likelihood of model divergence, a common issue observed in traditional Reptile-based outer-loop updates. Furthermore, transformer-based feature extraction is incorporated to enrich the informational content derived from limited data. To maintain computational efficiency, only the KL, COS, and DTW values are considered for weight computation. However, this approach can be extended to include other task-correlation metrics depending on the nature of the training data — for instance, a self-attention mechanism can be used to compute relevance weights in image-based tasks, though such extensions are beyond the scope of this work. The total number of training tasks is represented by N.

$$\Phi = \sum_{j=1}^{i-1} \left(W_{T_j} \Phi j \right) \tag{14}$$

$$W_{T_j} = \alpha \sum_{j=1}^{i-1} D_{KL(T_i,T_j)} + \beta \sum_{j=1}^{i-1} D_{COS(T_i,T_j)} + \gamma \sum_{j=1}^{i-1} D_{DTW(T_i,T_j)} \tag{15}$$

$$\begin{cases} D_{KL}(T_i, T_j) = \sum_k T_i(k) \ln T_i(k)/T_j(k) \\ D_{COS(T_i, T_j)} = 1 - \frac{\sum_j T_i T_j}{\sqrt{\sum_i T_i^2}\sqrt{\sum_j T_j^2}} \\ D_{DTW(T_i, T_j)} = D(u, v) = (q_u, c_v)^2 + \min\left[D(u-1, v), D(u, v-1), D(u-1, v-1)\right] \end{cases} \tag{16}$$

$$\Phi \leftarrow \Phi + \sum_{i=1}^{n}[0.1(1 - \frac{i}{N})(\widetilde{\Phi}_i - \Phi)] \tag{17}$$

4.1.3 Features from multi heads and layers of transformer encoder

The integration of transformers [66] plays a critical role in this study, as their capability to process long input sequences and model long-term dependencies is essential for a meta-RL agent to infer the underlying Markov Decision Process (MDP) from observed trajectories. Furthermore, the self-attention mechanism inherent to transformer architectures allows them to operate effectively even in environments characterized by sparse or noisy reward signals [46]. To maximize the information value of limited demonstration data, the training inputs are directly processed using BERT [60], a pre-trained transformer model trained with masked language modeling (MLM) and next sentence prediction (NSP) on large-scale corpora. However, determining the optimal number of attention heads and transformer layers for small-data training remains a significant challenge in this context. To address this issue, the outputs generated from multiple combinations of heads and layers provided by Hugging Face [63] are utilized as inputs to the Meta-IRL network. This approach leverages BERT's representational power to capture diverse aspects of the data and facilitates the analysis of how different architectural configurations affect performance under limited data conditions.

Figure 17 illustrates the procedure for integrating the pre-trained "BERT-base-uncased" model into the system. This model, with approximately 84M parameters, is first employed to tokenize the input corresponding to the current task. After tokenization, embedding layers are applied to encode the tokens and incorporate positional information, representing their relative or absolute order within the sequence. After this step, N transformer encoder layers are constructed to process the embedded input. Each encoder layer operates by projecting the query, key, and value representations within the multi-head attention mechanism into three separate feature spaces, after which scaled dot-product attention is applied to these projections. This iterative process continues across the stacked encoder layers, allowing the model to refine its representations progressively. The use of multiple attention heads further enhances representational capacity by enabling simultaneous focus on different aspects of the input. For instance, one attention head may focus on syntactic structure, while another emphasizes contextually significant tokens, thereby improving the model's ability to extract rich, meaningful information from the sequence.

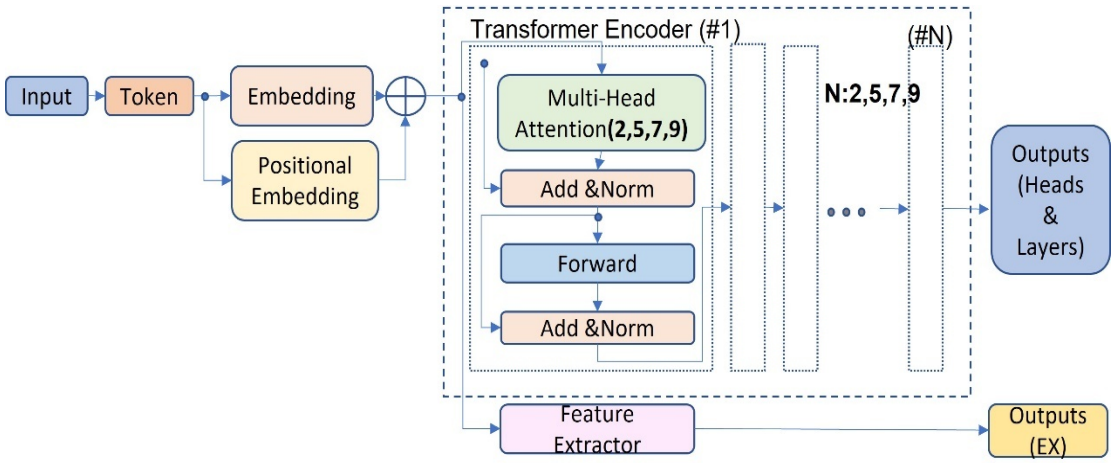


Figure 17. Multi heads and layers of transformer encoder combinations.

*Layers: {2, 5, 7, 9} and Heads: {2, 5, 7, 9}; Internal feature extractor (EX).

Given the emphasis on training under data-limited conditions, this study investigates how the number of multi-head attention units and transformer encoder layers affects the performance of a transformer-based translation model. To conduct this analysis, transformer encoder architectures with different configurations of layers {2,5,7,9} and multi-heads {2,5,7,9}are implemented and evaluated. Given the limited dataset size, preliminary evaluations indicated that increasing the number of heads beyond 10 yielded negligible performance gains. Consequently, head configurations ranging from 2 to 9 were selected for further analysis. In contrast to conventional practices that typically consider only even headcounts, both even and odd head numbers were deliberately included, as different head configurations may capture distinct semantic patterns under low-resource conditions. Representative layers exhibiting meaningful variation were then identified to ensure that the analysis reflects genuine differences rather than numerical artifacts of head selection. In addition, for comparative purposes, the output of the internal feature extractor (EX) “facebook/wav2vec2-base-960h” is incorporated into the

experimental setup. This inclusion enables a systematic evaluation of how different combinations of attention heads and encoder layers affect overall model performance.

4.2 Experiments

The experimental evaluation assessed the effectiveness of the proposed approach using data collected from the Applied Behavior Analysis (ABA) therapy process. The experiments were organized into four main components. First, as described in Section 4.2.3, the influence of two different internal configurations of hidden layers was examined. The objective was to enhance model performance with a low-computation strategy by adjusting the number of multi-heads and encoder layers, rather than increasing hidden layer sizes.

Next, in Section 4.2.4, the effectiveness of the task-oriented weight update mechanism was validated. Initially, the impact of various input features was analyzed to demonstrate the benefits of using representations derived from different multi-head and layer (H–L) combinations of the transformer encoder. The baseline input in this evaluation was the internal feature extractor (EX) from the pre-trained "facebook/wav2vec2-base-960h" model, as shown in Section 4.2.4.1. In Section 4.2.4.2, performance was further validated across single and multi-task scenarios by comparing EX inputs with those generated from different H–L feature configurations. This part aimed to verify the effectiveness of the proposed method, which integrates selected H–L features with task-oriented weighting.

Finally, in Section 4.2.5, new tasks were introduced to investigate the performance of models trained with different H–L feature configurations based on task-related weights.

The results from these experiments were analyzed to provide recommendations for the optimal number of multi-heads and encoder layers during both the training and testing phases.

4.2.1 Dataset

Interaction data between behavior technicians (BTs) and children diagnosed with Autism Spectrum Disorder (ASD) were collected during *wh*-question teaching sessions conducted at a university-based ABA clinic [5]. Each session was initiated by a *wh*-question presented by the BT, followed by a corresponding response from the child. The children's responses were then simplified and classified into four categories of consequences based on the professional assessment of their behavioral reactions by the BTs, represented as (R_C: {0, 1, 2, 3}):

- 0: Social praise for a correct answer.
- 1: Error correction for incorrect answers.
- 2: Prompts for no response.
- 3: Other situations, such as when the child cries, gets distracted, throws things, talks to themselves, or tries to run away.

The teaching process for each child was considered complete once the child answered the *wh*-question correctly on three consecutive occasions. A total of 19 distinct *wh*-question answering tasks were conducted, with the questions restricted to the categories *who*, *what*, *when*, and *where*. The dataset was split into 13 training tasks and 6 test tasks. In each task, a single interaction pair between the behavior technician (BT) and the child, represented as $SB(t)$ and $SC(t)$, was treated as the current state $State(t)$. Each task

consisted of multiple sequential interaction pairs *State*(1), *State*(2), . . . , *State*(*t*). The objective was to train the model to predict the appropriate action $Action_t$ according to the policy learned from demonstrations of the 13 training tasks $T_1, T_2, ..., T_{13}$. For each task, $Action_t$ denotes the next suitable response from the BT based on the historical states observed during the interaction sequence *State*(1), *State*(2), ..., *State*(*t*) of the current task.

After applying Google's speech-to-text technology to the audio data of the behavior technician (BT), the corresponding text sequences were obtained, as illustrated in Figure 18. These sequences were then processed using the transformer encoder described in the previous methodology to generate *SB*(*t*) with two types of output features: EX and head–layer combinations. The children's responses, denoted as $SC(t) \in R_C$, were represented as four distinct categories. For example, the response vector *SC*(1) was expanded to [1, 1, 1, 1, 1, 1, 1] to encode the "type 1" category in a vectorized form, which facilitates balance between the BT's input and the children's response data.

The model input, referred to as the current state *State*(*t*) (highlighted in the orange box), consisted of the most recent 14 interaction states [*s*(*t* - 13), *s*(*t* - 12), ..., *s*(*t*)]. Each state s(t)s(t)s(t) was defined as a pair [*SB*(*t*), *SC*(*t*)] (depicted in the blue box in the second column), representing the BT's utterance and the corresponding child response. Each state was encoded as a matrix with dimensions [$Max\ L * 24$, 14], which can be adjusted according to specific application requirements. In the experiments, *Max L* was set to 40, indicating the maximum sequence length of *SB*(*t*), while the number 24 referred to the hidden size H24 of the BERT encoder utilized in the model. Configurations with

12 and 24 hidden layers were compared to assess performance. To construct the current State(t), a window of 14 previous interaction states was considered in the input sequence.

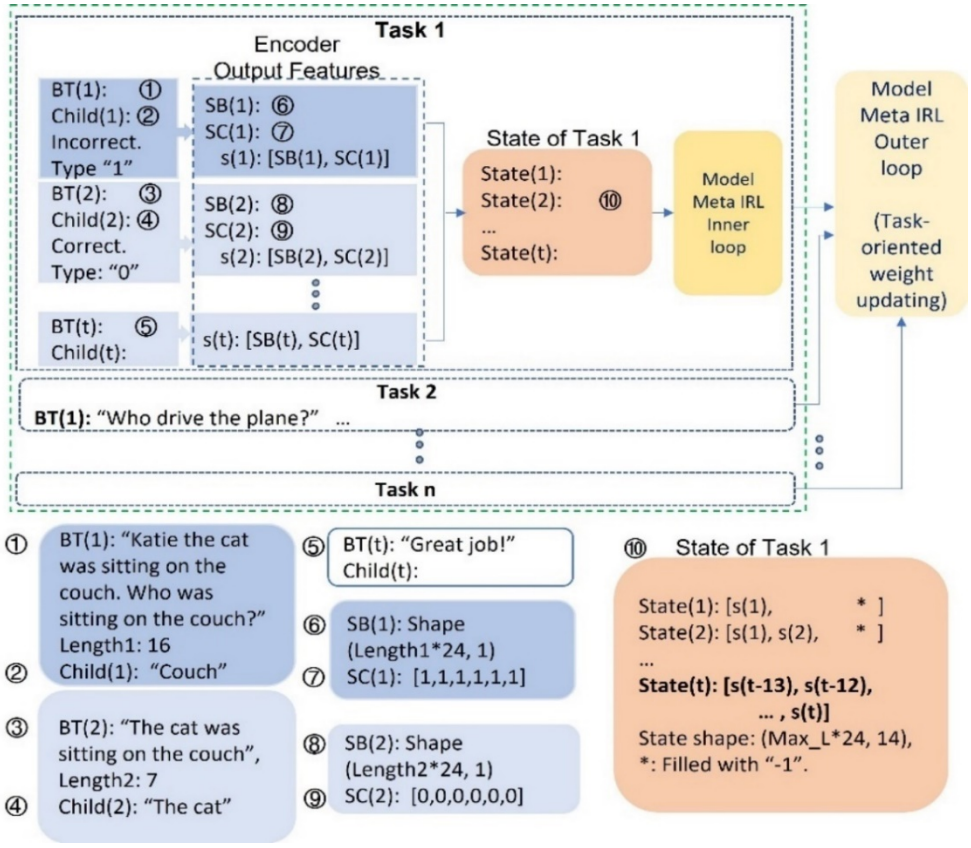


Figure 18. ABA clinic task-related data description.

* The outer loop processes all tasks with task-oriented weight updating; the inner loop processes State(t) for each task, where State(t) comprises 14 s(t) historical conversation pairs between BT and child s(t): [SB(t),SC(t)].

4.2.2 Evaluation metrics and hypotheses

When training data are limited, evaluating model performance solely on the distance between predicted trajectories and expert demonstrations can be unreliable because of variations in sentence structure or document subject between training and testing. For instance, a model might generate predictions that closely align with expert sentence structures yet deviate substantially in semantic content. To address this limitation, a

composite evaluation metric that integrates three distance measures— D_{COS}, D_{KL}, and D_{DTW} —along with their normalized forms G_{COS}, G_{KL}, G_{DTW}, and a constant term, is employed. As described in Equation (18), the cosine similarity (COS), Kullback–Leibler divergence (KL), and dynamic time warping (DTW) distances are defined as in Equation (16). The combined use of these three metrics mitigates the risk of relying on a single evaluation criterion and captures complementary aspects of trajectory similarity. Additionally, random coefficients R_1, R_2, and R_3 are introduced as proportional weights to balance the contribution of the three similarity components, with their sum constrained to equal 1.

Given the limited sample size, a Shapiro-Wilk test for normality and a chi-square test for distribution assessment were first applied to the data. Subsequently, Bootstrap paired *t*-tests and Mann-Whitney *U*-tests were conducted in the experiments to evaluate the proposed hypotheses. These hypotheses assess the distance between the model's predicted trajectory and the expert trajectory under different configurations. For example, H24 denotes a configuration with 24 inner layers in BERT, whereas 5H–2L denotes a setup with 5 attention heads in the multi-head attention mechanism and 2 transformer encoder layers.

- **H1:** Distance is reduced during training based on 5H- 2L and 5H-9L when utilizing H24 instead of H12.
- **H2:** Distance is reduced when utilizing the H-L features (5H-2L, 5H-5L, 5H-7L) compared to utilizing EX features.

- **H3:** Distance is reduced during training when utilizing a task-oriented Meta-IRL network compared to not using it.
- **H4:** Distance is reduced during new tasks testing when utilizing a task-oriented Meta-IRL network compared to not using it.

$$D = R_1 G_{KL}(D_{KL(T_P,T_E)}) + R_2 G_{COS}(D_{COS(T_P,T_E)}) + R_3 G_{DTW}\left(D_{DTW(T_P,T_E)}\right) + Cons \quad (18)$$

4.2.3 Impact of Bert-hidden layers on different numbers of transformer encoder layers

For each task, the input states $\{S_1, S_2, ..., S_t\}$ were first processed by the pre-trained transformer model ("BERT-base-uncased"). Configurations combining five attention heads (5H) with different numbers of encoder layers {2, 5, 7, 9} were evaluated, producing four transformer encoder outputs: 5H-2L, 5H-5L, 5H-7L, and 5H-9L. Because the maximum BERT hidden size of 768 significantly increases computational costs and because too few hidden layers may negatively impact learning under limited data conditions, model configurations with 12 and 24 hidden layers were specifically compared while accounting for hardware constraints.

The results presented in Figure 19 (a) and (b) demonstrate that, after training across 13 different tasks (x-axis) using features from the 5H-2L and 5H-9L transformer encoders, the model with 24 hidden layers consistently outperformed the 12-layer model, as measured by trajectory distance metrics. Additionally, the results indicate that as the number of training tasks increases, the performance of the 9L configuration continues to improve and begins to converge, whereas models using either H12 or H24 do not converge when relying on the sparse features extracted from the 2L configuration.

Furthermore, Figure 19 (c) shows that the performance of the 5L model with 24 hidden layers is comparable to that of the 9L model with 12 hidden layers. This finding suggests that it is essential to adjust the number of hidden layers according to the length of the training data and the available hardware resources, ensuring an appropriate balance between computational efficiency and model effectiveness.

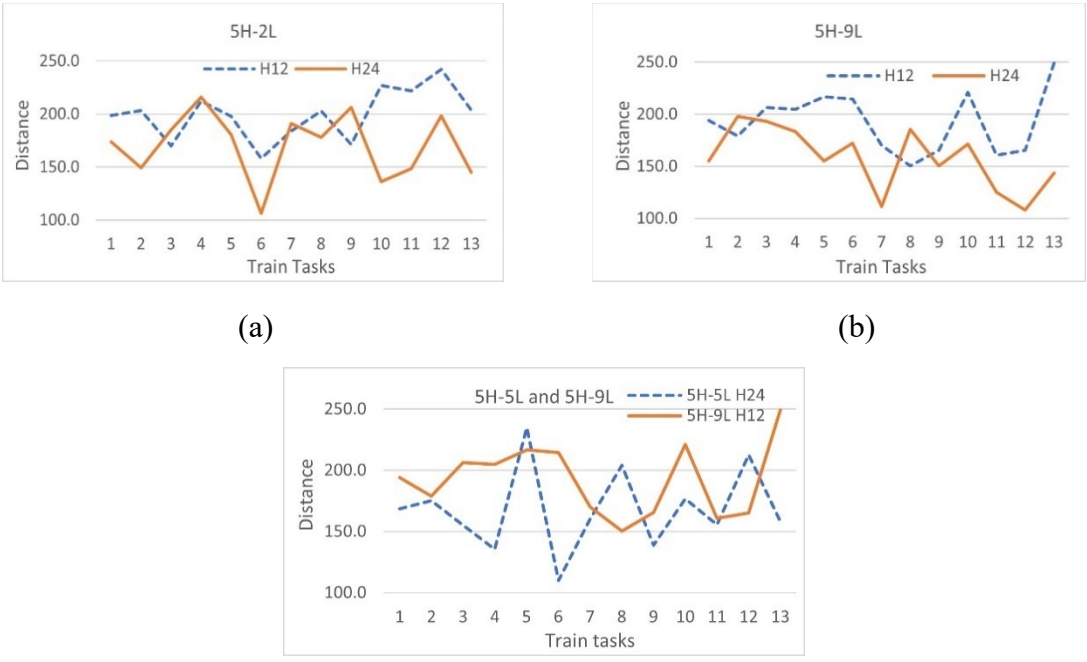


Figure 19. Training results based on 5H with different hidden layers.

As shown in Table 5, the t-test and Mann–Whitney (MW) tests indicate statistically significant differences as the number of hidden layers increases from H12 to H24. Specifically, the results for 5H-2L (ρ=0.026) and 5H-9L (ρ=0.016) both support Hypothesis H1. Although deeper architectures generally yield improved performance, they also introduce considerably higher computational costs. The experimental results further show that the performance of the 5L model with H24 is comparable to that of the 9L model with H12. This finding suggests that when the depth of feature extraction is

constrained by an insufficient number of hidden layers, increasing the number of encoder layers (e.g., from 2L to 9L) can be an effective strategy to enhance model performance.

Table 5. T-tests and Mann-Whitney tests for 5H with different hidden layers.

				T-test		MW-test
		Mean	**SE**	**t(12)**	**Sig.**	**Sig.**
5H-2L	12H	199.47	6.65	2.822	0.02	0.026
5H-2L	24H	170.37	8.68			
5H-9L	12H	192.12	8.17	3.385	0.002	0.016
5H-9L	24H	157.84	8.03			
5H-9L	12H	192.12	8.17	1.787	0.095	0.05
5H-5L	24H	168.15	9.34			

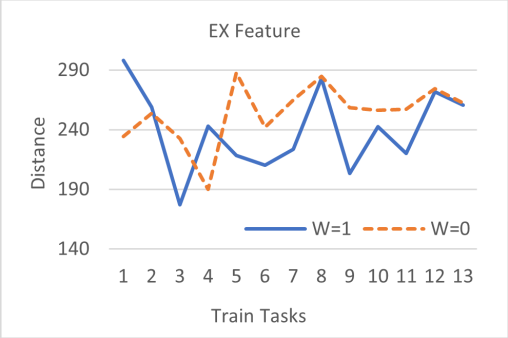
4.2.4 Task-oriented weight updating experiments

Three experimental evaluations were conducted to verify the effectiveness of the proposed task-oriented weight update mechanism. The first evaluation involved training the model on 13 tasks using **EX** as the input feature and testing its performance on 6 tasks, both with and without the task-oriented updating strategy. The second evaluation compared the training results for an **individual task** when using either **EX** or **H–L** (features derived from different numbers of attention heads and encoder layers) as inputs, again with and without the task-oriented update method applied. Finally, the third evaluation examined performance differences in a multi-task learning setup when employing EX and H–L input features, with and without incorporating the task-oriented weight updating approach.

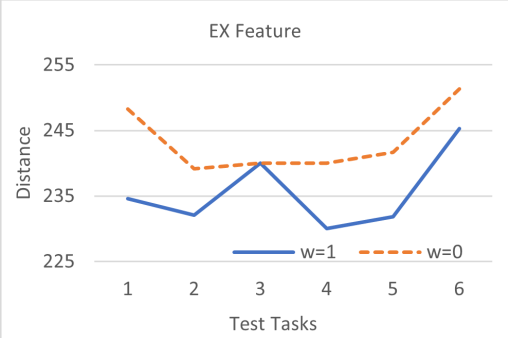
4.2.4.1 EX features as input

As illustrated in Figure 20, the model's performance was compared under two weight update strategies: the task-oriented approach (W=1) and the non-task-oriented approach (W=0). According to the results summarized in Table 6, there was no significant difference between the two methods across the initial six training tasks (x-axis). However, as the number of training tasks increased, the task-oriented method (W=1) began to demonstrate clear advantages. In the later tasks, the distance (y-axis) between the model's predicted trajectory and the expert trajectory was significantly smaller when W=1.

Moreover, the larger fluctuations observed in the W=1 results for specific relevant tasks (e.g., Task 8 and Task 12) indicate that the task-oriented update strategy can more effectively reinforce task-specific parameter adjustments, thereby improving adaptability to new tasks. The evaluation of six test tasks, presented in Figure 20 (b) and Table 6, further confirmed the superiority of the task-oriented approach. Although the W=1 configuration exhibited greater variability during training, it consistently yielded stronger performance outcomes. The test results (ρ=0.022) shown in Figure 20 (b) indicate that the task-oriented update method outperformed the non-task-oriented baseline, thereby supporting **Hypothesis H2** based on EX feature inputs.



(a)



(b)

Figure 20. Training and testing results based on EX.

Table 6. T-tests and Mann-Whitney tests for EX feature.

				T-test		MW-test
	W	Mean	SE	t(n)	Sig.	Sig.
EX-train	0	253.72	7.07	t(12)=-1.292	0.211	0.287
EX-train	1	239.30	9.58			
EX-test	0	243.40	2.08	t(5)=-4.099	0.022	0.04
EX-test	1	235.64	2.39			

4.2.4.2 Multi Heads and Layers features

1) Individual task

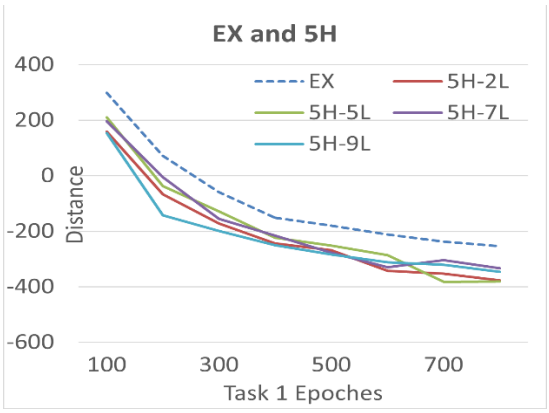
For an individual task (Task 1), the results presented in Figure 21 and Table 7 demonstrate that using EX as the input feature results in inferior performance compared to utilizing multiple heads and layers from the transformer encoder. Specifically, Figure 21 (a) shows that increasing the number of encoder layers (5H configurations: 5H-2L, 5H-5L, 5H-7L, 5H-9L) produced no significant improvement in training outcomes for a single task. In contrast, as depicted in Figure 21 (b), increasing the number of attention

heads (5L configurations: 2H-5L, 5H-5L, 7H-5L, 9H-5L) led to performance gains during training. The predicted outputs of the model did not follow a normal distribution when trained with 5H-9L and 9H-5L configurations (as indicated by asterisks in Table 7), and these cases were therefore evaluated using Mann–Whitney tests. According to Table 7, statistically significant differences (ρ=0.00) were observed for 5H-2L, 5H-5L, and 5H-7L, supporting Hypothesis H2 and indicating that H–L inputs outperform EX features, except in the cases of 5H-9L and 9H-5L.

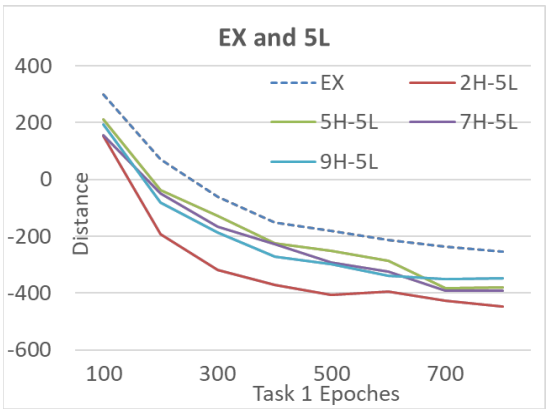
Nevertheless, selecting the optimal number of attention heads and encoder layers remains a challenge requiring further investigation. It is noteworthy that increasing the number of heads and layers (from 2L to 9L and from 2H to 9H) did not significantly influence training results for an individual task. This finding suggests that while multi-head and multi-layer configurations provide richer representational features, not all extracted features contribute equally to performance. Simply increasing the number of heads and layers (e.g., 9H, 9L) does not necessarily guarantee improved outcomes for a single-task setting.

2) Multiple tasks

In the multi-task experiments, the use of EX and H–L configurations (5H-2L, 5H-5L, and 5H-7L) as input features for training with the task-oriented update method (W=1) produced distinct outcomes. For each of the thirteen tasks, three independent training



(a)



(b)

Figure 21. Training results of Task1 with different heads and layers, EX feature.

Table 7. T-tests and Mann-Whitney tests for the individual task.

Task 1	W=1			T-test		MW-test
		Mean	SE	t(7)	Sig.	Sig.
	EX	-490.76	67.2			
EX and 5H	5H-2L	-608.24	63.82	17.004	0.00	-
	5H-5L	-585.28	70.14	9.128	0.00	-
	5H-7L	-578.35	66.10	13.274	0.00	-
	5H-9L*	-612.93	57.37	7.949	0.00	0.105
	2H-5L	-700.45	70.90	14.819	0.00	-
EX and 5L	7H-5L	-610.68	66.55	13.534	0.00	-
	9H-5L*	-610.36	187.78	18.832	0.00	0.083

rounds were conducted in a random sequence to minimize potential effects on contextual relationships between tasks and their subsequent results. The final prediction outcomes were obtained by averaging the predictions generated by the three trained models. This procedure enabled validation of the effectiveness of the task-oriented updating approach

and a comparison of the relative performance between the two types of input features. As shown in Figure 22, when EX features were used, the distance between the model's predicted trajectory and the expert trajectory did not converge, even as the number of training tasks increased. In contrast, when the model was trained with the 5H-2L input configuration, the trajectory distance gradually decreased as training progressed. According to Table 8, results obtained with different numbers of heads and layers (5H-2L, ρ=0.001; 5H-5L, ρ=0.000; 5H-7L, ρ=0.000) significantly enhanced model performance. These findings indicate that using H–L input features (5H-2L, 5H-5L, 5H-7L) yields superior results compared to EX features and support Hypothesis H2 in the context of multi-task learning with W=1.

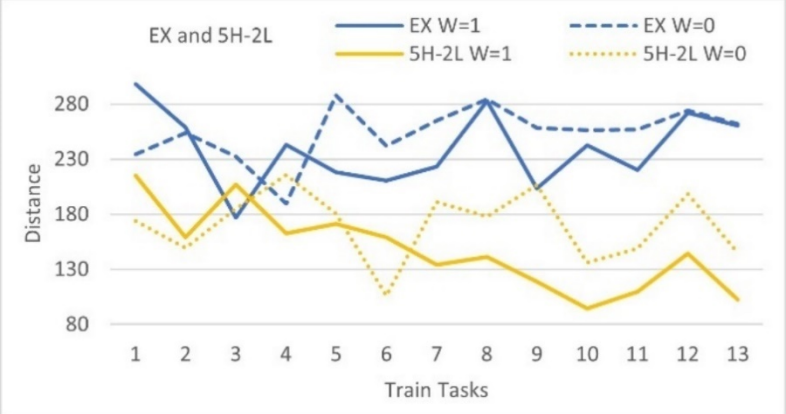


Figure 22. Training results based on EX and 5H-2L with W=0/1.

Next, using five attention heads (5H) and various transformer encoder layer configurations (2L, 5L, and 7L), the impact of the two update methods was examined under multi-task training conditions. As illustrated in Figure 23 and Table 9, analysis of the training results for the first half of the tasks indicates that neither the number of encoder layers nor the choice of update method (W=0 or W=1) produced significant

differences in model performance. However, as training progressed and more tasks were learned, the task-oriented approach (W=1) began to show clear advantages in the later tasks. This trend suggests that during the initial training phase, the model undergoes fluctuations as it adapts to different tasks, and only after exposure to several tasks does it begin to generalize and adjust effectively to new ones.

Table 8. T-tests for multi-task training (W=1).

			T-test	
	Mean	**SE**	**t(n)**	**Sig.**
EX	239.30	9.58	t(12)=6.563	0.001
5H-2L	147.66	10.27		
EX	239.30	9.58	t(12)=6.645	0.000
5H-5L	146.46	9.14		
EX	239.30	9.58	t(12)=6.518	0.000
5H-7L	149.01	7.76		

In Table 9, the predicted outputs did not follow a normal distribution when the model was trained with EX as the input feature for the final seven tasks (marked with asterisks), and these cases were therefore evaluated using the Mann–Whitney test.

Finally, as presented in Table 10, although there were only six test tasks and the output results for these tasks followed a normal distribution, additional Mann–Whitney (*MW*) test values were provided as references alongside the *t*-test results (EX cases marked with asterisks). All statistical results obtained from both the *t*-tests and MW tests indicate significant differences ($\rho<0.05$) between the compared groups, except for the EX

feature. This outcome supports Hypothesis H4, demonstrating that the application of the task-oriented approach enhances model performance in the testing tasks.

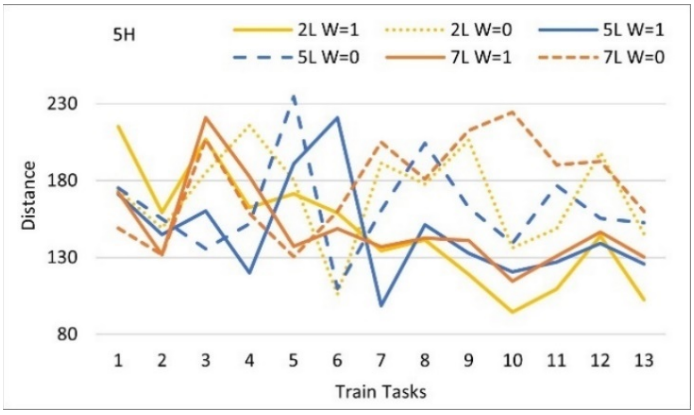


Figure 23. Training results based on 5H and three different layers (2,5,7) with W=0/1.

However, as shown in Figure 23, when the input features 5H-2L, 5H-5L, and 5H-7L were used, the experimental results indicated that although increasing the number of encoder layers can enrich the extracted information, it does not lead to a significant improvement in overall performance. This finding reinforces the view that selecting the optimal number of attention heads and layers remains challenging, especially under limited training data. It is therefore essential to account for the specific characteristics of the training tasks when determining the most appropriate model configuration.

4.2.5 Training and testing with different numbers of multi heads and layers of the transformer encoder

To address the challenge of selecting the appropriate number of attention heads and encoder layers (H–L), a series of experiments was conducted to determine the optimal configuration for both the training and testing phases. Six tasks were employed to evaluate the performance of sixteen trained models with varying H–L combinations,

Table 9. T-tests and Mann-Whitney tests (W=0/1 for multi-task training).

					T-test		MW-test
			Mean	SE	t(12)	Sig.	Sig.
Whole 13 tasks	EX*	W=0	253.72	7.07	1.292	0.235	0.287
		W=1	239.30	9.58			
	5H-2L	W=0	170.37	8.68	1.933	0.079	0.091
		W=1	147.66	10.27			
	5H-5L	W=0	162.60	8.65	1.304	0.280	0.113
		W=1	146.46	9.14			
	5H-7L	W=0	177.14	8.59	2.428	0.036	0.019
		W=1	149.01	7.76			
Last 6 tasks	EX	W=0	265.37	3.97	2.585	0.041	0.209
		W=1	243.57	11.16			
	5H-2L	W=0	172.03	10.69	7.692	0.023	0.002
		W=1	120.78	7.42			
	5H-5L	W=0	164.47	7.90	5.352	0.012	0.002
		W=1	127.97	6.20			
	5H-7L	W=0	195.12	8.07	5.981	0.002	0.001
		W=1	134.79	4.05			

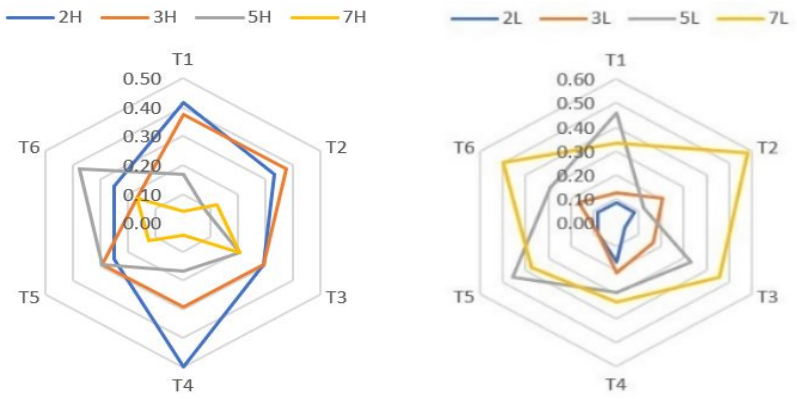
including 2H-(2L, 3L, 5L, 7L), 3H-(2L, 3L, 5L, 7L), 5H-(2L, 3L, 5L, 7L), and 7H-(2L, 3L, 5L, 7L). In scenarios with limited training data, configurations with H≥9 or L≥9

Table 10. T-tests and Mann-Whitney tests (W=0/1 for multi-task testing).

				T-test		MW-test
		Mean	**SE**	**t(5)**	**Sig.**	**Sig.**
EX*	W=0	259.28	0.140	1.163	0.297	0.818
	W=1	228.91	0.268			
5H-2L	W=0	258.19	0.069	5.185	0.004	0.002
	W=1	257.55	0.111			
5H-5L	W=0	258.08	0.195	4.571	0.006	0.026
	W=1	257.36	0.185			
5H-7L	W=0	257.82	0.205	2.722	0.042	0.041
	W=1	226.66	0.476			

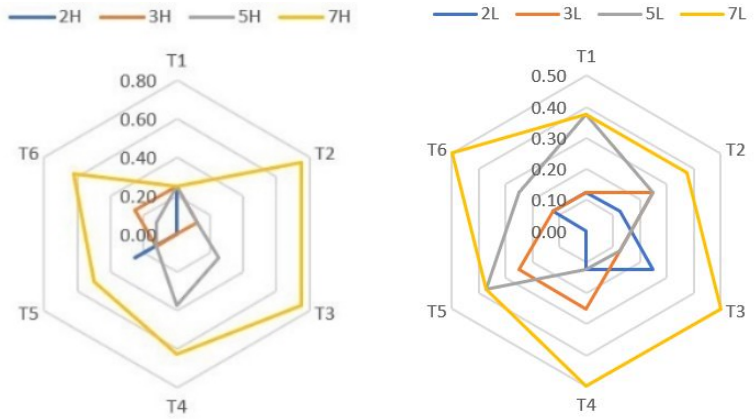
were excluded, as they introduced excessive information that caused the network to become overly sensitive, leading to instability or non-convergence.

Based on the prediction results of the top two models, the frequency distributions of heads and layers for each model were analyzed. As shown in Figure 24, the results suggest that lower values of H (2H, 3H, 5H in Figure 24 (a)) combined with moderately higher L values (5L, 7L in Figure 24 (b)) are more suitable for the training phase. For the testing phase, sixteen H–L feature configurations were used as inputs for six test tasks, and the frequency distributions of the top two models were similarly analyzed. The findings presented in Figure 25 indicate that higher H values (7H in Figure 25 (a)) and non-minimal L values (excluding 2L, as shown in Figure 25 (b)) yield better performance in the testing stage.



(a) (b)

Figure 24. H and L frequency distribution of trained models.



(a) (b)

Figure 25. H and L frequency distribution of tested data.

Additionally, the results reveal an important observation: employing identical H–L feature configurations for both training and testing phases does not necessarily yield optimal performance due to inherent variability between the two. This discrepancy arises from differences in the structure, content, and sequence length of sentences in the test tasks, particularly under limited data conditions. For instance, during training, two

models were generated using 2H-5L and 2H-7L configurations. However, during testing, feature processing with 2H-5L, 2H-7L, and 7H-5L was applied as input. The findings showed that the prediction results produced by the model trained with the 7H-5L configuration achieved the highest performance, highlighting the need to tailor H–L settings specifically for the testing stage rather than relying solely on those used during training.

4.3 Results and Discussion

The experimental findings presented in this chapter highlight key factors that shape the effectiveness of reinforcement learning under data-scarce HRI conditions and provide deeper insights into how task-oriented weight update meta-inverse reinforcement learning and transformer-based feature representations jointly influence model performance. The results demonstrate how architectural choices — including hidden layer depth, multi-head attention configuration, and task-oriented weight updating — interact with limited expert demonstrations to improve feature extraction, representation learning, and downstream task generalization in multi-task scenarios.

First, increasing network depth consistently enhances the model's predictive capability: the 24-layer architecture significantly outperformed its 12-layer counterpart. Nevertheless, this improvement is not unconditional. The number of hidden layers must be carefully matched to both the size of the training dataset and the available computational resources. Striking an appropriate balance is essential to ensure computational efficiency without compromising model effectiveness. Moreover, when representational capacity is constrained by insufficient depth, expanding the encoder

architecture (e.g., from 2L to 9L) is an effective strategy to improve feature extraction and downstream performance.

The experiments also demonstrate that feature selection and task orientation fundamentally affect the model's learning performance. When EX features were used as inputs, the presence or absence of task-oriented mechanisms made little difference during the initial six tasks. However, as the number of tasks increased, the task-oriented strategy began to exhibit clear advantages. A similar pattern was observed when using five attention heads (5H) with various encoder depths (2L, 5L, and 7L): while early training stages were characterized by performance fluctuations as the model adapted to diverse tasks, task-oriented learning (W = 1) ultimately facilitated more effective generalization as exposure increased. This transition reflects a broader phenomenon in multi-task learning — an initial phase of instability and variance, followed by a gradual emergence of cross-task knowledge transfer once the model encounters a sufficient range of tasks.

The comparative analysis of feature representations provides further evidence for this trend. Across most configurations, H–L inputs outperformed EX features, with the exception of 5H-9L and 9H-5L, indicating that richer hierarchical representations generally provide greater utility for generalization. Statistical analyses using both t-tests and Mann–Whitney tests further confirmed significant performance differences, reinforcing the conclusion that task-oriented design improves performance on downstream tasks. However, increasing encoder depth alone does not guarantee substantial gains: although deeper models extract more informative representations, configurations such as 5H-2L, 5H-5L, and 5H-7L showed limited performance

improvements. This suggests the presence of diminishing returns beyond a certain architectural depth, highlighting the importance of balancing representational richness with computational overhead.

Finally, a more nuanced understanding emerges from the analysis of attention-head and encoder-layer distributions. For training, architectures with fewer heads (lower H) and deeper layers (higher L) tend to yield better results, likely because deeper representations support more effective feature abstraction from limited training samples. In contrast, testing performance benefits from configurations with higher H and non-minimal L, which may enhance the model's ability to capture diverse, task-specific cues. Importantly, using identical H–L configurations in both training and testing does not necessarily produce optimal outcomes, reflecting inherent distributional and objective differences between these two phases. Notably, the model trained with a 7H-5L configuration achieved the highest predictive accuracy, emphasizing the importance of tailoring architectural settings specifically for the testing stage rather than directly transferring those used during training.

In summary, by selectively exploiting multi-head and multi-layer transformer features within a task-oriented Meta-IRL framework, the model learns more effective task-specific policies and generates more appropriate following responses compared to conventional Meta-IRL or single-layer baselines. Building on the response-recognition foundation established in Chapter 3, this chapter enables SARs to move beyond classification and toward adaptive, dialogue-driven interaction across different ABA task scenarios.

CHAPTER FIVE

IMPROVING OPTIMAL PROMPT LEARNING THROUGH MULTILAYER FUSION AND LATENT DIRICHLET ALLOCATION

After the model has learned to recognize a child's response (Chapter 3) and determine the appropriate action or dialogue type (Chapter 4), the final step toward autonomous SAR-assisted ABA therapy is to generate effective and context-aware responses that can help guide the child toward the learning objective. However, real ABA therapy differs significantly from generic NLP applications: the dialogue must be accurate, unambiguous, supportive, and therapeutically meaningful. Although Chapter 4 enables the system to generate the following response, the generated utterances still exhibit a substantial gap compared with those produced by human BTs in terms of clarity, instructional value, and emotional appropriateness. Therefore, improving the quality of generation is essential for achieving more natural, engaging, and therapeutically effective interactions in ABA settings. In practice, this challenge is further intensified by data scarcity. In ABA therapy, annotated demonstrations are limited and highly domain-specific, making it difficult for conventional prompt-based or few-shot learning methods to extract the nuanced features required for therapeutic dialogue. While few-shot learning with pre-trained language models has reduced the need for extensive fine-tuning, two persistent challenges remain: constructing high-quality prompts and overcoming data sparsity in specialized domains.

Motivated by this need, this chapter proposes a prompt-optimization framework that enhances generation quality through two complementary strategies. First, a Global

Attention Mechanism (GAM) is used to aggregate feature representations from multiple transformer layers, capturing rich semantic and structural cues that single-layer approaches often overlook. Second, Latent Dirichlet Allocation (LDA) topic features are incorporated as external semantic guidance to enrich prompt construction. Together, these strategies aim to generate responses that better align with therapeutic goals and more closely emulate the behavior of human BTs.

Extensive experimental evaluations on four benchmark datasets confirm that the proposed approach consistently outperforms state-of-the-art baselines, especially under low-resource conditions. These advantages are further highlighted in an applied behavior analysis (ABA) clinical dataset, where the proposed method significantly enhances therapeutic dialogue classification. Moreover, the optimized prompts produced in this chapter can serve as refined reference information for the response-generation model developed in Chapter 4, thereby further improving the clarity, instructional value, and overall quality of the BT's subsequent utterance. Parts of this chapter are adapted and expanded from the author's previous work [108].

5.1 Improving Prompt Learning based on the LM-RL framework

To further enhance prompt learning by effectively extracting additional information, this chapter extends beyond the conventional LM–RL framework and addresses a key limitation: when only the final-layer features of a language model (LM) are used as inputs for reinforcement learning (RL), crucial semantic information from earlier layers may be overlooked. To mitigate this issue, the proposed approach investigates multiple strategies for integrating LM features and incorporates complementary implicit

representations to optimize prompt generation. The methodology is organized into three main components: Section 5.1.1 outlines the fundamental LM–RL framework; Section 5.1.2 describes the fusion of multi-layer features from the language model to enrich representation learning, and Section 5.1.3 presents the integration of Latent Dirichlet Allocation (LDA) with LM features to improve prompt quality further.

5.1.1 Basic LM-RL framework

5.1.1.1 Bidirectional Encoder Representations from Transformers (BERT)

The architecture of BERT [60] is built on a multi-layer Transformer encoder [66], composed of multiple self-attention layers that capture and integrate information from various parts of an input sequence. The model undergoes pre-training on two core objectives: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, a subset of tokens in a sentence is randomly masked, and the model learns to predict them, thereby capturing contextual word dependencies. NSP, in contrast, is formulated as a binary classification task that determines whether a particular sentence logically follows another, enhancing the model's understanding of inter-sentential relationships.

BERT is released in two principal configurations: BERT-Base, consisting of 12 Transformer layers and approximately 110M parameters, and BERT-Large, which expands to 24 layers and 340M parameters. Beyond pre-training, BERT's architecture has been fine-tuned for a wide range of downstream natural language processing tasks, including sentiment analysis, text classification, named entity recognition, and question answering, demonstrating its versatility. RoBERTa [109] represents an optimized

extension of BERT and provides an in-depth replication study of its pre-training process while evaluating the impact of critical hyperparameters and data scale. Unlike BERT, RoBERTa employs a dynamic masking strategy that regenerates mask positions in each training iteration, thereby improving the model's capacity to capture contextual representations. Moreover, it removes the NSP objective entirely. It leverages a substantially larger corpus during training, enabling it to achieve state-of-the-art results on several benchmarks, including GLUE, RACE, and SQuAD.

5.1.1.2 Tempera

The effective utilization of large language models in zero-shot or few-shot settings relies heavily on well-designed prompts. Test-time prompt editing using Reinforcement learning (TEMPERA) [92] addresses this challenge by dynamically generating interpretable, query-specific prompts through test-time editing, formulated as a reinforcement learning problem. The framework features a flexible action space that enables modification of instructions, in-context examples, and verbalizers to tailor the prompt for a given query. TEMPERA uses the improvement in model performance after each edit as a reward signal and integrates optimization techniques, such as reward normalization, to enhance learning stability. Its attention-based policy architecture selectively prioritizes potential edits and design alternatives, demonstrating high effectiveness in reinforcement learning scenarios. Within this framework, three widely used forms of discrete prompts are typically considered for inclusion in the prompt pool:

- Instructions, which provide a segment of text describing how the task is performed, are usually put at the beginning.

- In-Context Demonstrations, which select several examples and their corresponding labels, are usually placed before the query.
- Verbalization, which aims to design how the task is asked and which keywords to select as labels.

The operational process proceeds as follows: during the testing phase, an initial prompt p_0 is first provided. TEMPERA then constructs a function f that accepts the initial prompt p_0, the query x, and a pool of examples/verbalizers p′ as inputs, producing a final optimized prompt defined as $p_f = f(p_0, x, p')$. It is important to note that the function f is trained on a fixed training dataset prior to deployment and can be directly applied during testing without additional training.

For reward formulation, TEMPERA adopts a score-difference-based strategy. For each query, the log-probability output from the language model using a prompt p_t with the correct label c is expressed as $\log(P_L(\hat{y}|x, p_t))$. The score difference $s(c,x,p_t)$ is defined as a combination of positive and negative components, represented respectively by $\lambda_1 \log(P_L(\hat{y}|x, p_t))$ and $-\lambda_2 \underset{c' \neq c}{\arg max} \log(P_L(\hat{y}_{c'}|x, p_t))$, where $\lambda_1 > 0$ and $\lambda_2 > 0$ are hyperparameters controlling the balance between the two terms.

While a traditional Markov Decision Process (MDP) emphasizes cumulative reward, prompt optimization focuses on the performance of the final prompt. Consequently, TEMPERA defines the immediate reward as the score difference between consecutive edits, calculated as $(s(c, x, p_t) - s(c, x, p_{t-1}))$. The reinforcement learning objective is to maximize this difference, thereby encouraging prompt updates that yield incremental performance improvements at each editing step.

5.1.1.3 LM-RL framework

As shown in Figure 26, the network architecture is constructed on the LM–RL framework. Within this framework, the language model (RoBERTa) is utilized for tasks such as text classification, while the reinforcement learning (RL) component provides prompt-level feedback to the language model. The final hidden-layer representations produced by the LM, together with the corresponding logits for each classification category, serve as the reward signal for RL-based prompt optimization.

Additionally, the final hidden-layer representations, along with intermediate feature vectors from earlier layers (H0, H1, H2), are further integrated through a Global Attention Mechanism (GAM) and Latent Dirichlet Allocation (LDA) to achieve enhanced feature fusion. This process produces a reinforcement learning (RL) state that is specifically optimized for prompt learning. With respect to the action space in the RL framework, and following a design similar to that of TEMPERA [92], the original prompts are derived from the Natural Instructions dataset [110], while the prompt templates are selected from the Prompt Source repository [111].

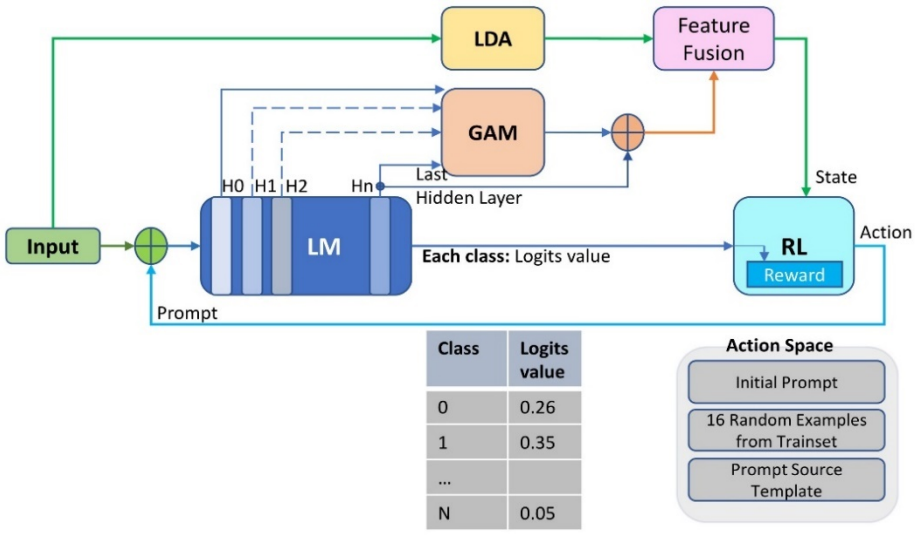


Figure 26. The LM-RL framework comprises three key components.

*LM for classification tasks, RL for prompt learning, and feature fusion outputs created by LDA and GAM, integrating input data and different layer features from LM. These fusion outputs serve as state inputs for RL-based prompt learning, while LM's classification logits provide reward. RL then generates prompt feedback to optimize the LM.

5.1.2 Fusion of Multilayer Features of Language Model

Drawing on the observations in Section 2.2.3 regarding the use of multiple layers in BERT, the primary objective is to recombine layer-wise representations so that the generalized features learned by the lower pre-trained layers are preserved, while the task-specific signals captured by the upper layers closer to the output are retained. This hierarchical integration strategy aims to leverage the complementary strengths of different abstraction levels within the language model. The proposed method is structured around two key steps designed to leverage further the information distributed across multiple layers of the LM:

First, as network depth increases, sparse but valuable features that remain in the lower layers of the language model (LM) when working with small datasets may become

suppressed or diluted in intermediate layers. To address this issue, deep hidden representations are individually extracted from different layers of the LM. Subsequently, the capabilities of the Global Attention Module (GAM) [112] are utilized to enhance feature integration. GAM operates simultaneously along the spatial and channel dimensions, thereby improving the comprehension of sequence representations. As a lightweight yet effective attention mechanism, GAM is designed to strengthen feature representations in both convolutional and transformer-based architectures. Explicitly modeling global contextual dependencies across spatial and channel dimensions enables the network to selectively emphasize informative features while diminishing the influence of less relevant ones.

Second, feature representations extracted from multiple hidden layers of the language model are integrated using GAM-based fusion. These representations are processed as three distinct GAM channels to capture high-level semantic consistency across transformer layers, thereby minimizing information loss under limited data conditions and enhancing the modeling of global contextual interactions. Specifically, GAM is applied to the concatenated hidden states derived from multiple transformer layers to refine the fused feature representation. This attention-guided fusion significantly improves the preservation of task-relevant signals, particularly in low-resource or noisy data scenarios.

As illustrated in Figure 27, the blue inputs on the left represent feature vectors extracted from different layers of the language model (denoted as H0, H1, Hn). Unlike the hierarchical processing of layers in the original language model, which is driven by

the attention mechanism, these layer-specific features are processed in parallel. The incorporation of GAM facilitates the exploration of interactions across these heterogeneous layers, enabling a more comprehensive integration of contextual information.

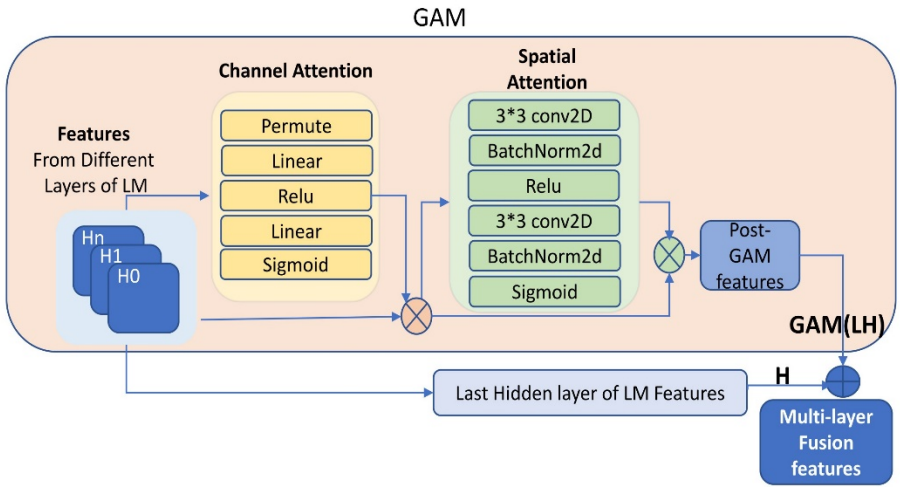


Figure 27. GAM fusion of features from multiple layers.

*GAM performs feature fusion across multiple layers of LM (H0, H1, Hn) to create GAM(LH). The final representation is formed by concatenating GAM(LH) with the LM's last hidden layer output H.

Following the application of GAM to various layers, additional feature representations are extracted, referred to as post-GAM features. Although these features originate from both the lower and final layers of the model, those derived from the lower layers may contain valuable information due to their sparsity and specificity. However, they can also introduce less relevant components, such as noise, which may diminish the usefulness of the shared representations in the final layer's output. To mitigate this issue, instead of directly utilizing the post-GAM features from individual layers, they are concatenated with the output vectors from the language model's final layer, a process denoted as GAM(LH) in Figure 27. For instance, feature vectors are extracted from both

the first hidden layer (L) and the final hidden layer (H), each with a shape of [32, 1, 1024]. These two vectors are initially combined to form a fused representation (LH), which is then processed by the GAM module to produce GAM(LH) (shape: [32, 1, 1024]). The resulting representation is subsequently added to the final-layer features (H) to generate the multi-layer fused output GAM(LH) + H (shape: [32, 1, 1024]). In this enhanced approach, GAM(LH) and H are treated as equally important contributors to the final fusion, ensuring that the common features from the language model's last layer are effectively reinforced.

This strategy not only integrates additional information from the post-GAM stage but also reduces the influence of potential noise present in those features, which might otherwise weaken the contribution of the final-layer representations. The concatenation step is designed to incorporate richer feature information while maintaining minimal computational cost, as it relies solely on a simple addition operation. Although more sophisticated fusion techniques could be employed to combine features from different layers, such methods would significantly increase computational complexity. Since the primary objective of this study is to demonstrate that further processing of multi-layer features can provide more effective information, a lightweight addition-based fusion strategy is adopted prior to applying GAM and integrating the post-GAM output with the H-layer. More computationally intensive fusion methods are reserved for future work. The overall structure of this process is depicted in Figure 27.

5.1.3 Latent Dirichlet Allocation (LDA) Fusion with LM

Latent Dirichlet Allocation (LDA) is a probabilistic text mining technique particularly well-suited for uncovering a small number of latent topics within large textual corpora. The model assumes that each document can be represented by a document–topic distribution (Θ), while each topic is characterized by a topic–word distribution (Φ). Given the complete set of words in a corpus, the primary objective of LDA is to infer the latent topic assignment (Z) for each word. Once these topic assignments are determined, the underlying topic distributions can be estimated as follows:

- Doc-topic distribution Θ: For each document, count how many words are assigned to each topic, and normalize to get the probability distribution. Example: In a document, 100 words belong to the topic Vision, and 200 to Language.
- Topic-word distribution Φ: For each topic, count how frequently each word appears under that topic to generate a word probability distribution. Example: In topic Vision, the word "image" appears 10 times, and "recognition" 20 times.

To estimate these distributions, the latent topic variable Z—which represents the topic assignment for each word—must first be determined. For instance, in the sentence *"Deep learning is powerful"*, a possible topic assignment might be Z = [Topic 3, Topic 3, Topic 2, Topic 3]. Since Z is not directly observable, a sampling-based approach is employed to approximate its posterior distribution P(Z|W), where W denotes the observed sequence of words. A widely used technique for this purpose is Collapsed Gibbs Sampling, which is frequently adopted in probabilistic and neural models with

stochastic components. This method generates samples of Z that approximate average scenarios across its potential values. The pseudocode below illustrates the process of applying Collapsed Gibbs Sampling in the context of LDA:

```
Algorithm: Collapsed Gibbs Sampling for LDA
Input:
   D : Document collection
   K : Number of topics
   α, β : Dirichlet priors
   V : Vocabulary size
   N_iter : Number of iterations
Initialize topic assignments z_dn randomly
Initialize count matrices: n_{d,k}, n_{k,w}, n_k
for iter = 1 to N_iter:
   for each document d in D:
      for each word position n in d:
         w = w_dn
         k = current topic z_dn
         # Remove current assignment
         n_{d,k} -= 1
         n_{k,w} -= 1
         n_k     -= 1
         # Compute conditional probabilities
         for t = 1 to K:
            p_t ∝ (n_{d,t} + α) × (n_{t,w} + β) / (n_t + V β)
         # Sample new topic
         z_dn = Sample(p_1, ..., p_K)
         # Update counts
         n_{d,z_dn} += 1
         n_{z_dn,w} += 1
         n_{z_dn}   += 1
Output:
   θ_{d,k} = (n_{d,k} + α) / (n_d + Kα)
   φ_{k,w} = (n_{k,w} + β) / (n_k + Vβ)
```

Building on the advantages outlined in Section 2.2.3 regarding the integration of language models and LDA, the outputs produced by LDA can serve as a basis for a wide range of downstream tasks, including document classification, similarity measurement,

and clustering. In this study, the topics generated by LDA offer complementary semantic information that differs from the representations typically captured by language model objectives. To exploit this advantage, LDA is utilized as a feature extraction mechanism, enabling the incorporation of latent semantic topic features into the output layer of the language model.

The implementation process primarily consists of training the LDA model to generate topic distributions and subsequently integrating these features into the overall framework. To determine the optimal number of topics, the LDA model is initially trained on the training subset of each dataset. Topic coherence is then computed following the approach described in [113], which has been shown to exhibit a strong correlation with human evaluative judgment. In particular, the Gensim library provides multiple metrics for assessing topic coherence, enabling a more comprehensive and reliable evaluation of the model's performance.

To improve the integration of outputs from the language model (LM) and LDA, the process begins with feature normalization. Directly merging these outputs can result in large-scale discrepancies, overgeneralization, and convergence difficulties [70]. In semantic NLP tasks, combining the output layers of LDA and LM can be interpreted as integrating distinct feature channels that represent the same input. Layer Normalization (LN) [114] is commonly recommended for this type of task. However, when dealing with small datasets, retaining the fine-grained details of both feature sets becomes crucial. In such scenarios, Instance Normalization (IN) [115] provides a more effective alternative.

Given the limited dataset conditions in this study, IN was adopted, although both LN and IN were evaluated during the LDA topic fusion experiments. This normalization strategy enhances the integration of semantic information between LDA and LM, producing richer and more informative features for the downstream reinforcement learning network. As illustrated in Figure 28, features from different LM layers are first processed through the GAM module. These features are then combined with the final hidden-layer representations to form multi-layer fused features, which are subsequently integrated with the LDA outputs before being fed into the RL network.

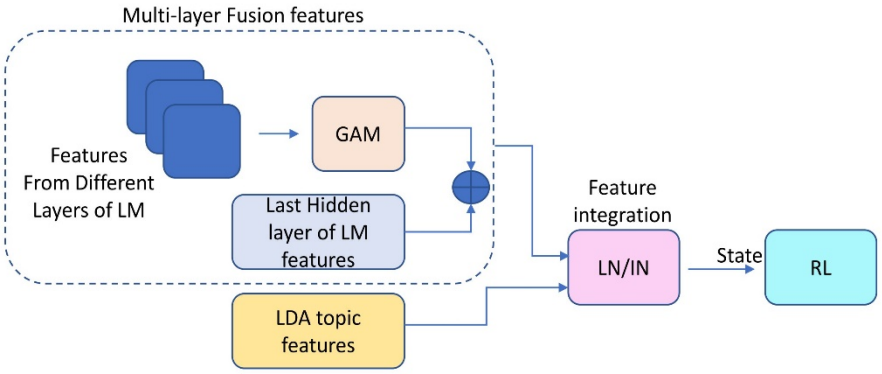


Figure 28. Multi-layer GAM fusion with LDA Integration.

*The multi-layer fusion features are combined with LDA topic features to form the final input for the RL network.

5.2 Experiments

In this section, building upon the LM–RL framework, two primary aspects are examined. The first focuses on assessing the impact of integrating specific language model layers into the GAM fusion process, while the second investigates the effects of incorporating LDA-based fusion alongside GAM. Following this, the proposed method is evaluated in comparison with eight state-of-the-art baseline models to assess its relative

effectiveness. In Section 5.2.3, an additional experiment is conducted to evaluate the performance of the proposed approaches on a dataset containing limited real-world data derived from a human–robot interaction (HRI) scenario.

5.2.1 Datasets, Baseline, and Experiment Setup

5.2.1.1 Datasets

The objective of the experiments is to validate the effectiveness of the proposed method across multiple datasets by evaluating its performance on few-shot text classification tasks. The evaluation encompasses sentiment analysis on single-sentence datasets such as SST-2 [116], MR [117], as well as multi-class classification on the AG_News [118] dataset. Beyond semantic analysis, the applicability of the approach is further examined in a specialized human–robot interaction (HRI) context. Specifically, similar to Section 4.2.1, a dataset containing interaction records between behavior technicians (BT) and children with Autism Spectrum Disorder (ASD) during *wh*-question teaching sessions is utilized for evaluation. This task is distinct from conventional classification or generation tasks due to its highly specialized context and the limited size of the dataset, as described in Section 5.2.3.

5.2.1.2 Baseline

The effectiveness of the proposed approach is assessed through comparative evaluation against a set of representative baseline methods. These baseline models include:

- Finetuning [60];
- Continuous prompt : Black-Box Tuning, AutoPrompt[60], [87] ;

- Discrete prompt: Manual Prompt [111], and In-Context Demonstration [119];
- RL prompt: RLPrompt [91], Tempera [92].

5.2.1.3 Experiment Setup

Text classification experiments were conducted using consistent configurations based on the TEMPERA framework to ensure a fair comparison across models. The proposed approach employed RoBERTa-large as the underlying language model and adopted a Proximal Policy Optimization (PPO) reinforcement learning structure. Initial instruction prompts were derived from Natural Instructions [110], while prompt templates were sourced from PromptSource [111]. Each task utilized 16 randomly selected training samples per category, forming a small-sample dataset suitable for prompt-based learning. Standard test sets from the baseline methods were used for performance reporting.

For topic generation, the LDA component was implemented using the Gensim library, which was applied to train topic models and extract latent topics for each dataset following topic coherence evaluation. For the HRI dataset, the same language model and reinforcement learning configuration were used. However, due to the dataset's unique characteristics and the high risk of overfitting under the standard 16-sample setting, the methodology was adjusted by modifying the input data format, adapting the PromptSource templates, and refining the LDA topic design, as described in Section 5.2.3.

5.2.2 Experiments on few-shot learning classification tasks

5.2.2.1 GAM on single layer experiments

This section presents a systematic investigation of how selecting different hidden layers from the language model (LM) influences the performance of the reinforcement learning (RL) network. Initial exploration of the lower layers (L0, L1, L2) showed minimal variation in results, which led to a more focused analysis of three representative layers: the first hidden layer (L, closest to the input), the middle hidden layer (M, the twelfth layer in RoBERTa), and the final hidden layer (H, nearest to the model output). As shown in Table 11, single-layer experiments revealed that layers L and M consistently delivered lower performance than layer H. This performance gap can be attributed to the structural properties of deep neural networks: lower layers tend to contain more redundant information, making them less suitable for fine-grained classification tasks.

Analysis of GAM Integration Across Network Layers: A comprehensive evaluation of GAM application at different network depths revealed distinct performance patterns. When applied to the lower layer (L), which contains abundant raw information, GAM significantly enhanced performance, improving accuracy from 59.1 to 74.2 on the SST-2 classification task. Moderate gains were observed at the middle layer (M), where performance increased from 81.2 to 85.0. In contrast, applying GAM to the final layer (H) decreased accuracy from 90.6 to 83.8. This degradation is likely caused by a “double-filtering” effect, as the final layer’s features have already undergone extensive refinement through the language model’s hierarchical processing, and further filtering can obscure essential information.

Overall, these findings indicate that GAM's effectiveness is inversely correlated with the depth of the layer to which it is applied. The most substantial improvements are achieved when GAM operates on information-rich lower layers, whereas its impact diminishes—or even becomes detrimental—when applied to pre-filtered higher-level representations.

5.2.2.2 GAM on two layers and enhanced Fusion experiments

The analysis of multi-layer feature fusion revealed intricate dynamics between representations derived from lower and higher layers. Two primary fusion strategies were examined: a direct combination of lower and higher layers (LH) and a GAM-enhanced fusion incorporating a softmax mechanism (GAM(LH), as shown in Table *11*). The experimental outcomes (LH = 64.6, GAM(LH) = 85.9) indicated that both fusion strategies underperformed compared to the single higher layer (H = 90.6). This counterintuitive result is primarily due to two factors: first, the inherent noise in lower-layer features, which can compromise the refined representations produced by the final layer; and second, the fusion mechanism's implicit bias toward higher-layer information.

Although combining complementary signals from multiple layers theoretically enhances feature richness and diversity, the evaluated fusion methods were unable to fully capitalize on the lower layers' feature space without diminishing the discriminative capacity of the higher layers. These observations suggest that achieving effective integration of multi-layer representations may require more advanced fusion techniques capable of balancing noise suppression with information preservation.

Table 11. GAM on different layers (Accuracy %) of four datasets.

		SST-2	AG_News	MR	Clinic	
Single Layer	L	59.1	65.1	80.7	21.9	Single layer L
	M	81.2	68.3	83.2	31.3	Single layer M
	H	**90.6**	**80.3**	**88.5**	**53.1**	Single layer H
	GAM(L)	74.2	68.7	83.5	28.1	GAM on single layer L
	GAM(M)	85.0	73.2	86.8	40.6	GAM on single layer M
	GAM(H)	83.8	77.9	88.4	46.9	GAM on single layer H
Two Layers	LH	64.6	80.0	86.9	37.5	Direct combination of two layers LH
	GAM(LH)	85.9	73.1	87.4	50.0	GAM-enhanced fusion of two layers LH
	GAM(LH)+H	**91.6**	**82.4**	**88.9**	**56.3**	Concatenating H with GAM(LH)

Subsequent experiments refined the fusion strategy by integrating both GAM-processed multi-layer representations and the original high-level features. This enhanced configuration, referred to as GAM(LH)+H in Table 11, employs a two-phase fusion mechanism: initially applying the Global Attention Module (GAM) to the combined lower- and higher-layer features to generate GAM(LH), followed by the concatenation of the original higher-layer vector (H) with this processed output. The resulting architecture achieved notably improved performance (GAM(LH)+H: 91.6) over earlier approaches, highlighting the effectiveness of combining complementary information from different

network depths. The results show that this design preserves the refined features of higher layers while adding useful contextual signals from lower layers. This combination produces a more robust feature representation and highlights the benefit of hierarchical fusion in few-shot prompt learning.

5.2.2.3 Fusion of LDA features experiments

A comprehensive evaluation was conducted to determine the optimal number of topics for Latent Dirichlet Allocation (LDA), testing values from 10 to 200. The selection emphasized model simplicity and topic coherence, using the leftmost inflection point of the coherence curve as the optimal setting. For instance, 32 topics were chosen for the SST-2 dataset, despite the highest coherence appearing at 35. Across experiments, topic counts between 30 and 50 consistently produced stable results. After selecting topic numbers, separate models were trained for each dataset. Because LDA outputs often contain many zeros, the results were stored as sparse matrices to reduce memory consumption.

After generating topic features, the experimental analysis of LDA integration followed a two-stage design. First, the direct combination of LDA-derived topic vectors with the final layer output of the language model (H+LDA in Table 12) was examined. Building on this, an enhanced fusion architecture was introduced that merged GAM-based layer fusion (GAM(LH)+H) with LDA topic features ((GAM(LH)+H)+LDA in Table 12). To address heterogeneous data distributions and varying sample sizes, two normalization strategies—Layer Normalization (LN) and Instance Normalization (IN)—

were applied. This framework enabled a systematic evaluation of LDA's contribution while ensuring robustness across different data conditions.

Table 12. Fusion of LDA (Accuracy %).

	SST-2	AG_News	MR	Clinic	
H	90.6	80.3	88.5	53.1	Single layer H
H+LDA	90.2	79.4	85.8	53.1	Direct integration of final layer with LDA
(GAM(LH)+H)+LDA: LN	91.8	84.6	88.7	56.3	GAM-based multi-layer fusion with LDA
(GAM(LH)+H)+LDA: IN	**93.1**	**86.2**	**89.3**	**59.4**	GAM-based multi-layer fusion with LDA

5.2.2.4 Comparison with the baseline

The results in Table 13, show consistent performance improvements across all four datasets. The proposed method surpasses the best baseline accuracies by 0.6% on SST-2, 0.7% on AG_News, 1.3% on MR, and 6.3% on Clinic. Standard deviations were computed over multiple prompt sets (four for the Clinic dataset and three for each of the others), providing additional insight into result variability. This evaluation framework offers strong empirical evidence of the method's effectiveness in few-shot text classification and highlights its stability under different prompting strategies.

Table 13. Comparison with the baseline(Accuracy %).

	SST-2	AG_News	MR	Clinic
Finetuning	80.6(3.9)	84.9(3.6)	67.4(9.7)	43.8(2.6)
AutoPrompt	75.0(7.6)	65.7(1.9)	62.0(0.8)	40.6(4.4)
Black-Box Tuning	89.1(0.9)	93.2(0.5)	86.6(1.3)	-
Manual Prompt	82.8	76.9	80.9	53.1
In-Context Demo	85.9(0.7)	74.9(0.8)	80.6(1.4)	40.6(1.7)
RLPrompt	92.5(0.8)	80.2(0.7)	87.1(0.4)	46.9(2.3)
Tempera	91.9(2.0)	85.5(1.5)	88.0(1.1)	46.9(2.8)
This paper	**93.1(0.8)**	**86.2(1.0)**	**89.3(0.7)**	**59.4(1.8)**

5.2.3 ABA Clinic dataset experiment

Building on the LM-RL framework, the method was evaluated using a specialized HRI dataset collected from the same ABA clinic. This task poses significant challenges by requiring both response evaluation and strategic response generation. BTs must interpret the child's behavior within context and deliver appropriate responses based on therapeutic goals and prior interaction history. Additional complexity arises from the limited number of demonstrations and the variability in dialogue decomposition—conversations rarely follow a fixed structure and can differ widely even for identical tasks. As a result, direct fine-tuning or prompt-based generative approaches are often ineffective. Furthermore, the most similar pre-training dataset [120], which focuses on sentence-pair causal reasoning in domains such as blogs and photography, is poorly

aligned with the clinical objective of generating the next therapeutic utterance from historical dialogue rather than predicting simple causal links.

To address the lack of suitable pre-training data, a strategy inspired by the SWAG dataset architecture [121] was adopted. In SWAG, a context is provided, and the model must predict the most probable continuation from four options. Following this structure, each dialogue instance was reformulated as a concatenated sequence pair ("current sentence + four candidate next sentences"), enabling the language model to perform feature extraction and multi-class classification. Due to the small dataset size (118 training and 32 test samples), a hybrid prompt engineering approach was applied. This included SWAG-based templates ("appropriatecontinuation," "howends," "firstthen," "firstthenkey") combined with custom-designed templates ("nextsentence," "afterthissentence," "firstthenpredictnext"). Additionally, LDA features trained on four datasets were incorporated to enhance contextual representation and improve classification performance.

The experiments were structured into several parts: direct language model fine-tuning, various prompt-based methods (AutoPrompt, Manual Prompt, In-Context Demo), and LM-RL approaches (RLPrompt, Tempera, and the proposed LM-RL architecture). Initial attempts to integrate LDA features using topics generated solely from the clinic dataset yielded minimal improvements, mainly due to limited data size and insufficient topic diversity. In such cases, the model requires richer topic diversity rather than alternative modeling techniques applied to the same data. To address this limitation, additional experiments utilized LDA topics generated from a larger, combined dataset

comprising four different sources. This approach introduced no task-specific overlap with the clinic data while significantly enriching topic diversity. The goal was to assess the effectiveness of incorporating LDA-derived topics; future work could further optimize topic selection through broader cross-dataset comparisons.

The results demonstrated clear performance gains, highlighting the benefit of supplementing low-resource datasets with more diverse topic representations. Performance metrics—summarized in Table *11* to Table 13 and illustrated in Figure 29 to Figure 31—include confusion matrix accuracy (0.541, 0.552, 0.553, 0.589), F1-weighted

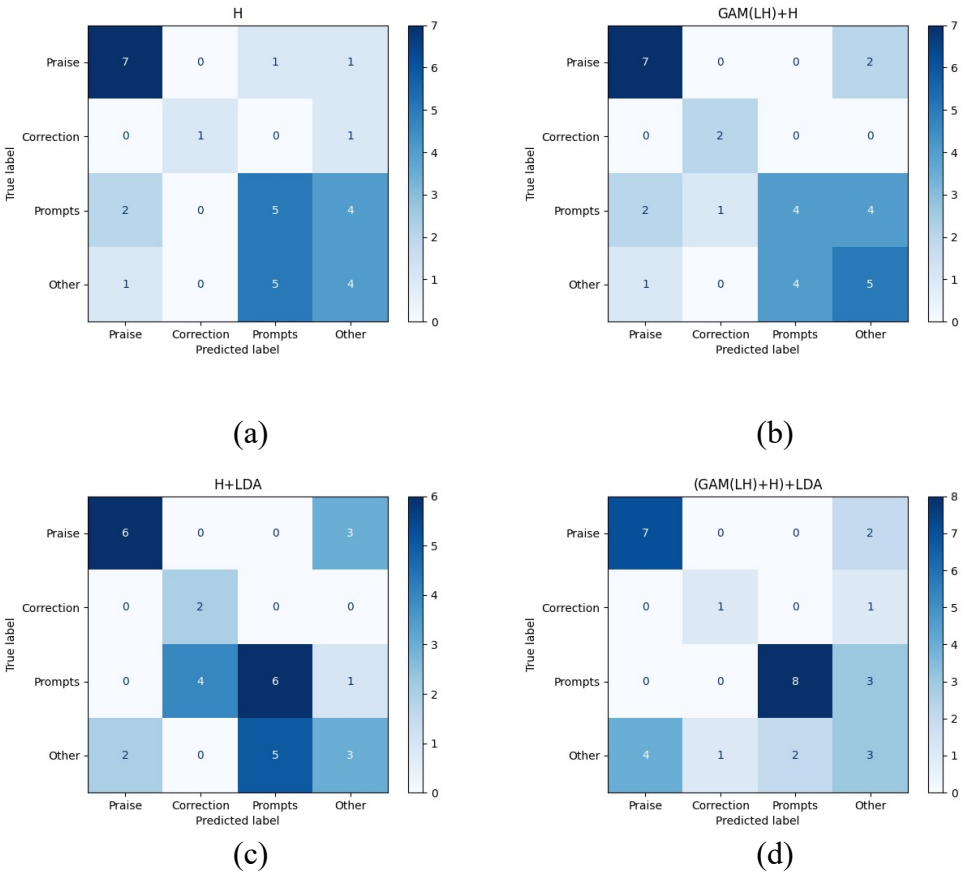


Figure 29. Confusion matrix of four fusion features.

* (H, GAM(LH)+H, H+LDA, (GAM(LH)+H)+LDA).

scores, and precision values, providing comprehensive evidence of the approach's effectiveness.

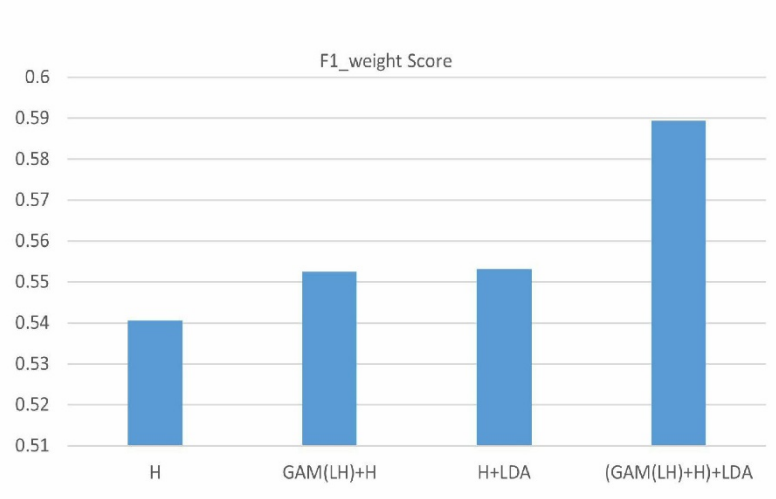


Figure 30. F1_weighted score of four fusion features of the clinic dataset.

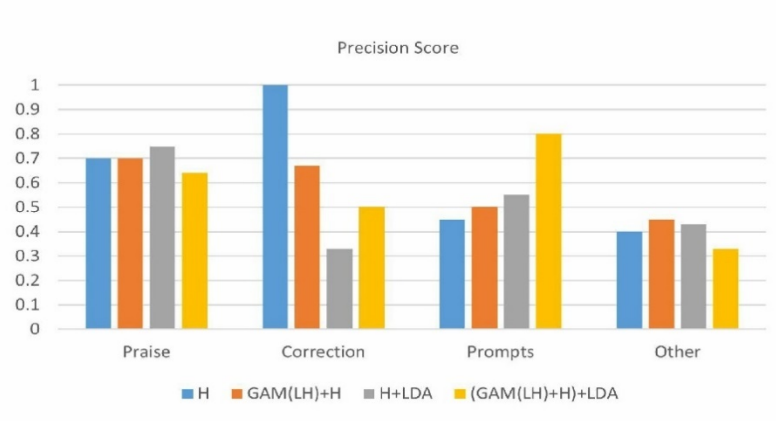


Figure 31. Precision score in four categories.

* (Praise, Correction, Prompts, Other) of the clinic dataset.

5.3 Discussion and Conclusion

The GAM(LH)+H configuration achieves superior performance in few-shot learning (Table 11) by treating the H-layer vector and GAM(LH) output as equally essential components. This design captures additional useful features often missed by conventional hierarchical processing, making it particularly effective for prompt learning under limited-data conditions.

A deeper examination of feature integration results (Table 12) indicates that while the direct fusion of the final-layer output with LDA topic vectors (H+LDA) provides only marginal improvements over the baseline, the enhanced architecture employing GAM-based fusion combined with Instance Normalization ((GAM(LH)+H)+LDA) achieves substantially greater performance gains. This outcome demonstrates the importance of sophisticated integration strategies in effectively exploiting complementary information beyond that captured by the standard language model.

The comparison of results from Table 11 and Table 12 further reinforces this conclusion. The output derived solely from the final layer (H: 90.6) serves as a robust baseline, reflecting the inherent strength of high-level semantic representations in language models. However, the improved configuration that leverages multi-layer fusion via the GAM(LH)+H design achieves 91.6% accuracy, underscoring the advantage of aggregating features across multiple network depths. In contrast, a direct combination of LDA-generated topic vectors with the final-layer output (H+LDA: 90.2) fails to surpass the baseline, highlighting the limitations of naive feature concatenation without structured fusion.

Significantly, when an advanced integration approach is employed—one that merges GAM-driven hierarchical feature extraction with LDA-derived topic representations ((GAM(LH)+H)+LDA)—the model exhibits further performance improvements. This is attributed to the complementary strengths of the two components: GAM enhances the representation of contextual and structural information distributed across multiple transformer layers, while LDA contributes higher-level semantic signals that enrich the model's understanding of latent topic distributions. The synergy between these two modules enables the network to capture a more diverse and informative set of features, ultimately leading to stronger downstream performance and improved generalization in few-shot learning tasks.

The analysis of the ABA clinic dataset results (Figure 29 to Figure 31) demonstrates consistent performance gains in both accuracy and F1-weighted scores, particularly within the "Prompts" category. The confusion matrix (Figure 29) indicates that the model achieves strong classification performance in the "Praise" and "Prompt" categories when employing GAM-based multi-layer fusion combined with LDA, reflecting the framework's ability to capture and utilize relevant contextual features for these response types. However, the "Correction" and "Other" categories show higher misclassification rates, suggesting that these classes present additional challenges due to data scarcity and semantic variability. The F1-weighted scores further support this observation, with the GAM-LDA fusion method yielding the highest values overall, demonstrating an effective balance between precision and recall for "Praise" and "Prompt." Precision analysis provides additional insight: while "Praise" and "Prompt" achieve comparatively higher

precision, “Correction” and “Other” remain considerably lower, reflecting greater classification uncertainty.

Performance variation across categories is closely linked to data distribution. “Praise” and “Prompt” maintain robust results in part due to their relatively larger number of training samples and clearer semantic boundaries. By contrast, “Correction” is limited by insufficient sample size, constraining the model’s ability to generalize. The “Other” category poses the most significant challenge, primarily due to the inherent complexity of the data. In this case, all responses that did not fall into the three primary categories were aggregated into “Other,” resulting in a highly heterogeneous class that included diverse, semantically inconsistent samples. This broad variability reduces class separability and leads to a higher rate of misclassification, underscoring the importance of more granular labeling or data augmentation strategies for improving performance in such complex categories.

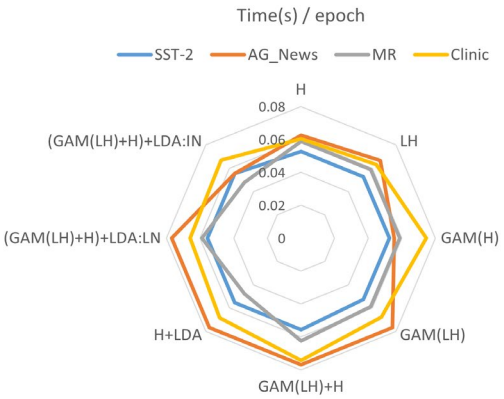


Figure 32. Time(s) of each epoch in four datasets.

*Single final layer (H), two layers (LH), GAM-enhanced final layer (G(H)), GAM-enhanced two layers (G(LH)), concatenating the H layer with the two layers GAMenhanced fusion output (GAM(LH)+H),

direct integration of final layer with LDA (H+LDA), GAM-based multi-layer fusion with LDA (GAM(LH)+H)+LDA), dual normalization strategies: LN and IN.

Across all evaluations, LM-RL-based approaches consistently outperform non-LM-RL baselines, with the proposed method achieving higher accuracy and lower standard deviations, as reported in Table 13. Although Black-Box Tuning exhibits slightly better performance on the AG-News four-class classification task, the LM-RL framework surpasses all prompting-based baselines in overall effectiveness and stability. Figure 32 presents the training speed of various methods (measured in seconds per epoch) across four datasets. Due to the relatively small size of the Clinic dataset, its results were normalized to ensure scale consistency within the figure. The results reveal that incorporating GAM into different layers introduces an increase in computational cost. However, subsequent inclusion of H-layer features does not increase the feature vector length, and using 32 topics in the LDA module contributes only minimally to feature dimensionality. Consequently, these additions do not significantly affect training speed.

Training times across most datasets—SST-2, MR, and Clinic—remain comparable across different methods. The primary exceptions are observed in the AG-News dataset, where the training times of (GAM(LH)+H)+LDA with Layer Normalization (LN) and Instance Normalization (IN) diverge more noticeably. This difference likely stems from AG-News text representations being more sensitive to normalization strategies, underscoring the importance of deeper feature exploration and adaptive design when applying advanced fusion methods to datasets with distinct linguistic or semantic characteristics.

In summary, this chapter introduces a prompt-optimization framework that integrates multi-layer semantic fusion and topic-guided enhancements. Experimental results indicate that, given the interaction history between the BT and the child, the proposed method achieves strong performance in predicting the next response category within ABA therapy sessions. Within the full ABA interaction pipeline, the output of this framework can serve as a complementary enhancement to the response-generation model developed in Chapter 4, providing refined reference information that further improves the clarity, instructional value, and overall quality of the BT's subsequent utterances, thereby narrowing the gap between SAR-generated responses and those of human BTs.

CHAPTER SIX

SUMMARY

6.1 Conclusion

In Chapter 3, to understand the child's response under limited demonstration conditions, techniques such as Self-Imitation Algorithm (SIA), Auxiliary Classifier GAN (ACGAN), dynamic experience replay (ER) resizing, and hierarchical DQN are employed to enhance data efficiency in reinforcement learning. Unlike existing approaches that primarily focus on optimizing sample storage and retrieval structures, the proposed framework introduces a dual-DQN architecture in which DQN2 performs targeted reprocessing based on DQN1's output decisions. Experimental analyses using two datasets indicate that this method effectively improves performance without increasing the number of demonstration samples.

The applicability of these techniques extends beyond the current context, offering potential benefits for domains where data are sparse, limited, or exhibit high inter-sample similarity. The validation conducted in this study focuses on human verbal and nonverbal audio recognition using limited data—an essential component in training social assistive robots (SARs) to deliver personalized ABA interventions. As the primary decision-making entity in HRI, SARs leverage this recognition capability to guide subsequent decision-making processes based on human behavioral responses. The hierarchical network is built on DQN with an RL architecture, which can also leverage its strengths in sequential decision-making.

In Chapter 4, to decide and adapt its next action based on task context, the proposed approach leverages a pre-trained network trained on large-scale datasets as an encoder for feature extraction and integrates a task-oriented Meta-IRL network with externally updated parameters to enhance adaptability to new tasks. To maximize the utility of limited training data, various transformer configurations with different numbers of heads and layers are systematically explored, and practical guidance is provided for selecting the most effective settings. Experimental results based on limited robot-mediated therapy demonstrations confirm that this method significantly improves reinforcement learning performance under data-constrained conditions.

In Chapter 5, to optimize prompt generation and thereby produce therapeutically effective language that aligns with ABA teaching goals, a novel framework is proposed that integrates Global Attention Mechanism (GAM) with multi-layer feature representations and LDA-based topic incorporation, achieving enhanced feature extraction in few-shot learning scenarios. The effectiveness of this approach derives from the strategic integration of hierarchical features through GAM and the enrichment of the feature space with topic-level semantics provided by LDA. Consistently strong performance across multiple datasets confirms the framework's versatility and demonstrates its potential to optimize prompt generation and improve learning efficiency in data-limited environments.

6.2 Future Work

Future research can build upon this work in several directions. First, a key area for advancement lies in improving the understanding of how behavior technicians (BTs)

make decisions based on complex, dynamic interaction states. Future studies will explore training strategies for increasingly complex tasks, such as treating different therapy objectives and programs as distinct tasks. Additionally, the current assessment of children's responses in discrete trial training (DTT) relies heavily on professional judgment, posing challenges for learning from limited data. Developing automated or semi-supervised evaluation methods could help overcome this limitation and enable more scalable applications.

Second, one limitation of the current framework is its reliance on layer-specific feature vectors, which may reduce its adaptability to diverse language model architectures. Automating task-specific feature selection for prompt optimization—especially in relation to human intent, environmental context, and engagement modeling—remains an essential challenge and a promising area for further research.

Third, the analysis of the ABA clinic dataset revealed methodological constraints, particularly when transforming therapeutic interactions into a multiple-choice classification format. These limitations were most evident in the "Other" category, which aggregated heterogeneous interaction types. Future work will focus on more granular data categorization and on the use of fine-tuned generation models trained on ABA therapy scripts to better capture the complexity of therapeutic interactions while maintaining computational efficiency.

Finally, expanding the framework to address broader aspects of social assistive robotics (SARs) represents a significant future direction. Potential efforts include enabling robots to learn high-level interaction rules from multimodal cues—such as pitch,

intensity, and speaking rate—and automating personalized tasks like tracking patient healthcare data, adjusting treatment plans, and coordinating care with healthcare professionals. Future research will investigate adaptations to a broader range of real-world HRI tasks.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” Annu. Rev. Control Robot. Auton. Syst., vol. 3, no. 1, May 2020, pp. 297–330.

[2] W.-Y. G. Louie and G. Nejat, “A learning from demonstration system architecture for robots learning social group recreational activities,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea: IEEE, Oct. 2016, pp. 808–814.

[3] A. Hijaz, J. Korneder, and W.-Y. G. Louie, “In-the-wild learning from demonstration for therapies for autism spectrum disorder,” in 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN), Aug. 2021, pp. 1224–1229.

[4] W.-Y. G. Louie, J. Korneder, A. Hijaz, and M. Sochanski, “Investigating therapist vocal nonverbal behavior for applications in robot-mediated therapies for individuals diagnosed with autism,” in Social Robotics, Lecture Notes in Computer Science, vol. 12483, A. R. Wagner et al., Eds. Cham: Springer, 2020, pp. 416–427.

[5] W.-Y. G. Louie, J. Korneder, I. Abbas, and C. Pawluk, “A study on an applied behavior analysis-based robot-mediated listening comprehension intervention for ASD,” Paladyn, Journal of Behavioral Robotics, vol. 12, no. 1, Jan. 2021, pp. 31–46.

[6] H. Liu, Y. Chen, J. Tan, and M. H. A. Jr, “Improving learning from demonstrations by learning from experience,” arXiv:2111.08156, Nov. 2021.

[7] T. Hester et al., “Deep Q-learning from demonstrations,” Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[8] B. Kang, Z. Jie, and J. Feng, “Policy optimization with demonstrations,” in International conference on machine learning, PMLR, 2018, pp. 2469-2478.

[9] Y. Zhang, Y. Lan, Q. Fang, X. Xu, J. Li, and Y. Zeng, “Efficient reinforcement learning from demonstration via Bayesian network-based knowledge extraction,” Computational Intelligence and Neuroscience, vol. 2021, no. 1, p. 7588221, Jan. 2021.

[10] M. Vecerik, O. Sushkov, D. Barker, T. Rothorl, T. Hester, and J. Scholz, “A practical approach to insertion with variable socket position using deep reinforcement learning,” in 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada: IEEE, May 2019, pp. 754–760.

[11] Y. Wu, G. Tucker, and O. Nachum, “Behavior regularized offline reinforcement learning,” arXiv:1911.11361, Nov. 2019.

[12] L. C. Cobo, K. Subramanian, C. L. Isbell, A. D. Lanterman, and A. L. Thomaz, “Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains,” Artificial Intelligence, vol. 216, Nov. 2014, pp. 103–128.

[13] S.-A. Chen, V. Tangkaratt, H.-T. Lin, and M. Sugiyama, “Active deep Q-learning with demonstration,” Mach. Learn., vol. 109, no. 9–10, Sep. 2020, pp. 1699–1725.

[14] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 2018, pp. 6292-6299.

[15] S. Choi, K. Lee, and S. Oh, “Robust learning from demonstration using leveraged Gaussian processes and sparse-constrained optimization,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden: IEEE, May 2016, pp. 470–475.
[16] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, “Residual reinforcement learning from demonstrations,” arXiv:2106.08050, 2021.
[17] M. Clark-Turner and M. Begum, “Deep reinforcement learning of abstract reasoning from demonstrations,” in Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI), Chicago, IL, USA: ACM/IEEE, Feb. 2018, pp. 160–168.
[18] A. Tyshka and W.-Y. G. Louie, “Transparent learning from demonstration for robot-mediated therapy,” in 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Napoli, Italy: IEEE, Aug. 2022, pp. 891–897.
[19] S. R. Hosseini, A. Taheri, M. Alemi, and A. Meghdari, “One-shot learning from demonstration approach toward a reciprocal sign language-based HRI,” Int. J. of Soc. Robotics, vol. 16, no. 4, Apr. 2024, pp. 645–657.
[20] K. Li and J. W. Burdick, “Meta inverse reinforcement learning via maximum reward sharing for human motion analysis,” arXiv:1710.03592, Oct. 2017.
[21] Y. Guo, J. Gao, Z. Wu, C. Shi, and J. Chen, “Reinforcement learning with demonstrations from mismatched task under sparse reward,” in Conference on Robot Learning, PMLR, 2023, pp. 1146-1156.
[22] S. K. S. Ghasemipour, S. Gu, and R. Zemel, “SMILe: scalable meta inverse reinforcement learning through context-conditional policies,” Advances in Neural Information Processing Systems, 32, 2019.
[23] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in International conference on machine learning, PMLR, 2019, pp. 5331-5340.
[24] P. Wang, H. Li, and C.-Y. Chan, “Meta-adversarial inverse reinforcement learning for decision-making tasks,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi’an, China: IEEE, May 2021, pp. 12632–12638.
[25] Z. Xiong, L. Zintgraf, J. Beck, R. Vuorio, and S. Whiteson, “On the practical consistency of meta-reinforcement learning algorithms,” arXiv:2112.00478, Dec. 2021.
[26] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” Found. Trends Mach. Learn., vol. 11, no. 3–4, 2018, pp. 219–354.
[27] M. Wołczyk and A. Krutsylo, “Remember more by recalling less: investigating the role of batch size in continual learning with experience replay (student abstract),” AAAI, vol. 35, no. 18, May 2021, pp. 15923–15924.
[28] B. Hao, Y. Duan, T. Lattimore, C. Szepesvari, and M. Wang, “Sparse feature selection makes batch reinforcement learning more sample efficient,” in International Conference on Machine Learning, PMLR, 2021, pp. 4063-4073.

[29] B. Daley, C. Hickert, and C. Amato, “Stratified experience replay: correcting multiplicity bias in off-policy reinforcement learning,” arXiv:2102.11319, Feb. 2021.
[30] M. Andrychowicz et al., “Hindsight experience replay,” Advances in neural information processing systems, 30, 2017.
[31] W. Fedus et al., “Revisiting fundamentals of experience replay,” in International conference on machine learning, PMLR, 2020, pp. 3061-3071.
[32] R. Liu and J. Zou, “The effects of memory replay in reinforcement learning,” in 2018 56th annual allerton conference on communication, control, and computing (Allerton), IEEE, 2018, pp. 478-485.
[33] S. Zhang and R. S. Sutton, “A deeper look at experience replay,” arXiv:1712.01275, Apr. 2018.
[34] L. Zhang et al., “A framework of dual replay buffer: balancing forgetting and generalization in reinforcement learning,” in Proceedings of the 2nd Workshop on Scaling Up Reinforcement Learning (SURL), International Joint Conference on Artificial Intelligence (IJCAI), p. 88. 2019.
[35] A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov, and A. I. Panov, “Forgetful experience replay in hierarchical reinforcement learning from demonstrations,” arXiv:2006.09939, Jun. 2020.
[36] Q. Wei, H. Ma, C. Chen, and D. Dong, “Deep reinforcement learning with quantum-inspired experience replay,” IEEE Trans. Cybern., vol. 52, no. 9, Sep. 2022, pp. 9326–9338.
[37] X. Li et al., “Findings of the first shared task on machine translation robustness,” in Proc. 4th Conf. Machine Translation (Volume 2: Shared Task Papers, Day 1), Florence, Italy: Assoc. Comput. Linguistics, 2019, pp. 91–102.
[38] Q. Chen, E. Dallas, P. Shahverdi, J. Korneder, O. A. Rawashdeh, and W.-Y. G. Louie, “A sample efficiency improved method via hierarchical reinforcement learning networks,” in 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Napoli, Italy: IEEE, Aug. 2022, pp. 1498–1505.
[39] Y. Gui and P. Doshi, “Inversely learning transferable rewards via abstracted states,” arXiv:2501.01669, Feb. 2025.
[40] L. Zhou, C. Xu, and J. J. Corso, “Towards automatic learning of procedures from web instructional videos,” in Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1. 2018.
[41] T. Yu, P. Abbeel, S. Levine, and C. Finn, “One-shot hierarchical imitation learning of compound visuomotor tasks,” arXiv:1810.11043, Oct. 2018.
[42] H. Karnan, G. Warnell, F. Torabi, and P. Stone, “Adversarial imitation learning from video using a state observer,” in 2022 International Conference on Robotics and Automation (ICRA), IEEE, 2022, pp. 2452-2458.
[43] M. Jing et al., “Reinforcement learning from imperfect demonstrations under soft expert guidance,” in Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 04, 2020, pp. 5109-5116.

[44] S. Wang, K. Wei, H. Zhang, Y. Li, and W. Wu, “Let me check the examples: Enhancing demonstration learning via explicit imitation,” arXiv:2209.00455, Aug. 2022.
[45] D. Pathak et al., “Zero-shot visual imitation,” in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2018, pp. 2050-2053.
[46] L. Chen et al., “Decision transformer: reinforcement learning via sequence modeling,” Advances in neural information processing systems 34, 2021, pp. 15084-15097.
[47] L. C. Melo, “Transformers are meta-reinforcement learners,” in international conference on machine learning, PMLR, 2022, pp. 15340-15359.
[48] P. Michel, O. Levy, and G. Neubig, “Are sixteen heads really better than one?,” Advances in neural information processing systems 32, 2019.
[49] H. Gong, Y. Tang, J. Pino, and X. Li, “Pay better attention to attention: head selection in multilingual and multi-domain sequence modeling,” Advances in Neural Information Processing Systems 34, 2021, pp. 2668-2681.
[50] J. Li, R. Cotterell, and M. Sachan, “Differentiable subset pruning of transformer heads,” Transactions of the Association for Computational Linguistics 9, 2021, pp. 1442-1459.
[51] T. Ji, S. Jain, M. Ferdman, P. Milder, H. A. Schwartz, and N. Balasubramanian, “On the distribution, sparsity, and inference-time quantization of attention values in transformers,” in Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online: Association for Computational Linguistics, 2021, pp. 4147–4157.
[52] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, “On the effect of dropping layers of pre-trained transformer models,” Computer Speech & Language, vol. 77, p. 101429, Jan. 2023.
[53] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, “Revealing the dark secrets of BERT,” in Proc. EMNLP-IJCNLP, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 4364–4373.
[54] W. Ma, K. Zhang, R. Lou, L. Wang, and S. Vosoughi, “Contributions of transformer attention heads in multi- and cross-lingual tasks,” in Proc. ACL (Long Papers), 2021, pp. 1956–1966.
[55] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does BERT look at? An analysis of BERT’s attention,” arXiv:1906.04341, Jun. 2019.
[56] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, “Analyzing multi-head self-attention: specialized heads do the heavy lifting, the rest can be pruned,” arXiv:1905.09418, Jun. 2019.
[57] A. Rogers, O. Kovaleva, and A. Rumshisky, “A primer in BERTology: what we know about how BERT works,” Trans. Assoc. Comput. Linguistics, vol. 8, Dec. 2020, pp. 842–866.
[58] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, “On identifiability in transformers,” arXiv:1908.04211, 2019.
[59] L. Liu, J. Liu, and J. Han, “Multi-head or single-head? An empirical comparison for transformer training,” arXiv:2106.09650, Jun. 2021.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, Minneapolis, MN, USA: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.
[61] N. Zhang et al., “Differentiable prompt makes pre-trained language models better few-shot learners,” arXiv:2108.13161, May 2022.
[62] Y. Tang, S. Guo, J. Liu, B. Wan, L. An, and J. K. Liu, “Hierarchical reinforcement learning from imperfect demonstrations through reachable coverage-based subgoal filtering,” Knowledge-Based Systems, vol. 294, p. 111736, Jun. 2024.
[63] T. Wolf et al., “Transformers: State-of-the-art natural language processing,” in Proc. EMNLP 2020: System Demonstrations, Online: Assoc. Comput. Linguistics, 2020, pp. 38–45.
[64] W. De Vries, A. Van Cranenburgh, and M. Nissim, “What’s so special about BERT’s layers? A closer look at the NLP pipeline in monolingual and multilingual models,” in Findings of EMNLP 2020, Online: Assoc. Comput. Linguistics, 2020, pp. 4339–4350.
[65] Z. Zhang, S. Wu, D. Jiang, and G. Chen, “BERT-JAM: Maximizing the utilization of BERT for neural machine translation,” Neurocomputing, vol. 460, Oct. 2021, pp. 84–94.
[66] A. Vaswani et al., “Attention is all you need,” Advances in neural information processing systems 30, 2017.
[67] M. Jia et al., “Visual prompt tuning,” in European conference on computer vision, pp. 709-727. Cham: Springer Nature Switzerland, 2022.
[68] X. Liu et al., “P-Tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks,” arXiv:2110.07602, Mar. 2022.
[69] M. Venugopalan and D. Gupta, “An enhanced guided LDA model augmented with BERT-based semantic strength for aspect term extraction in sentiment analysis,” Knowledge-Based Systems, vol. 246, p. 108668, Jun. 2022.
[70] N. Peinelt, D. Nguyen, and M. Liakata, “tBERT: Topic models and BERT joining forces for semantic similarity detection,” in Proc. ACL 2020, Online: Assoc. Comput. Linguistics, 2020, pp. 7047–7055.
[71] M. Xiang, D. Zhong, M. Han, and K. Lv, “A study on online health community users’ information demands based on the BERT-LDA model,” Healthcare, vol. 11, no. 15, p. 2142, Jul. 2023.
[72] P. Zhang, H. Zhao, F. Wang, Q. Zeng, and S. Amos, “Fusing LDA topic features for BERT-based text classification,” preprint, Nov. 2022.
[73] J. Bragg, A. Cohan, K. Lo, and I. Beltagy, “FLEX: Unifying evaluation for few-shot NLP,” Advances in neural information processing systems 34, 2021, pp. 15787-15800.
[74] X. Liu et al., “GPT understands, too,” AI Open 5, 2024, pp. 208-215.
[75] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, “Conditional prompt learning for vision-language models,” in Proc. CVPR 2022, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 16795–16804.

[76] Y. Bai et al., “Training a helpful and harmless assistant with reinforcement learning from human feedback,” arXiv:2204.05862, Apr. 2022.
[77] Y. Zhou et al., “Large language models are human-level prompt engineers,” in The eleventh international conference on learning representations. 2022.
[78] A. Rajeswaran et al., “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations,” arXiv:1709.10087, Jun. 2018.
[79] Y. Zha, L. Guan, and S. Kambhampati, “Learning from ambiguous demonstrations with self-explanation guided reinforcement learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 9, 2024, pp. 10395-10403.
[80] A. Koubaa, “ROSGPT: Next-generation human-robot interaction with ChatGPT and ROS,” preprint, Apr. 26, 2023.
[81] M. Xu et al., “Prompting Decision Transformer for Few-Shot Policy Generalization,” in international conference on machine learning, PMLR, 2022, pp. 24631-24645.
[82] Y. Gu, X. Han, Z. Liu, and M. Huang, “PPT: Pre-trained Prompt Tuning for Few-shot Learning,” arXiv:2109.04332, Mar. 13, 2022.
[83] J. Wei et al., “Finetuned Language Models Are Zero-Shot Learners,” arXiv:2109.01652, Feb. 8, 2022.
[84] C. Yang et al., “Large Language Models as Optimizers,” in The Twelfth International Conference on Learning Representations. Sep. 7, 2023.
[85] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel, “Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution,” arXiv:2309.16797, Sep. 28, 2023.
[86] R. Anil et al., “PaLM 2 Technical Report,” arXiv:2305.10403, Sep. 13, 2023.
[87] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, “AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts,” in Proc. EMNLP 2020, Online: Association for Computational Linguistics, Nov. 2020, pp. 4222–4235.
[88] Y. Hao, Z. Chi, L. Dong, and F. Wei, “Optimizing Prompts for Text-to-Image Generation,” Advances in Neural Information Processing Systems 36, 2023, pp. 66923-66939.
[89] H. Sun, A. Hüyük, and M. van der Schaar, “Query-Dependent Prompt Evaluation and Optimization with Offline Inverse RL,” arXiv:2309.06553, Mar. 7, 2024.
[90] W. Kong, S. A. Hombaiah, M. Zhang, Q. Mei, and M. Bendersky, “PRewrite: Prompt Rewriting with Reinforcement Learning,” arXiv:2401.08189, Feb. 20, 2024.
[91] M. Deng et al., “RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning,” arXiv:2205.12548, Oct. 22, 2022.
[92] T. Zhang, X. Wang, D. Zhou, D. Schuurmans, and J. E. Gonzalez, “TEMPERA: Test-Time Prompt Editing via Reinforcement Learning,” arXiv:2211.11890, 2022.
[93] H. Jung and K.-J. Kim, “Discrete Prompt Compression with Reinforcement Learning,” IEEE Access 12, 2024, pp. 72578-72587.
[94] I. J. Goodfellow et al., “Generative Adversarial Networks,” Communications of the ACM 63, no. 11, 20120, pp. 139-144.

[95] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125-1134.
[96] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, “End-to-End Incremental Learning,” in Computer Vision – ECCV 2018, vol. 11216, Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 241–257.
[97] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” arXiv:1511.05952, Feb. 25, 2016.
[98] N. Holz, P. Larrouy-Maestri, and D. Poeppel, “The paradoxical role of emotional intensity in the perception of vocal affect,” Sci. Rep., vol. 11, no. 1, p. 9663, May 2021.
[99] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora, “Deep Reinforcement Learning for Tactile Robotics: Learning to Type on a Braille Keyboard,” IEEE Robotics and Automation Letters 5, no. 4, 2020, pp. 6145-6152.
[100] Q. Chen, J. Korneder, O. Rawashdeh, Y. Wang, and W.-Y. G. Louie, “Optimizing Reinforcement Learning with Limited HRI Demonstrations: A Task-Oriented Weight Update Method with Analysis of Multi-Head and Layer Feature Combinations,” preprint, 2025.
[101] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” arXiv:1509.02971, Jul. 5, 2019.
[102] G. Zuo, K. Chen, J. Lu, and X. Huang, “Deterministic generative adversarial imitation learning,” Neurocomputing, vol. 388, May 2020, pp. 60–69.
[103] A. Nichol, J. Achiam, and J. Schulman, “On First-Order Meta-Learning Algorithms,” arXiv:1803.02999, Oct. 22, 2018.
[104] L. Yu, T. Yu, C. Finn, and S. Ermon, “Meta-Inverse Reinforcement Learning with Probabilistic Context Variables,” Advances in neural information processing systems 32, 2019.
[105] K. P. Murphy, Machine Learning: A Probabilistic Perspective. Cambridge, MA: The MIT Press, 2018.
[106] A. Singhal, “Modern Information Retrieval: A Brief Overview,” IEEE Data Eng. Bull. 24, no. 4, 2001, pp. 35-43.
[107] O. Gold and M. Sharir, “Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier,” ACM Transactions On Algorithms (TALG) 14, no. 4, 2018, pp. 1-17.
[108] Q. Chen, J. Korneder, O. Rawashdeh, and G. Louie, “Improving Optimal Prompt Learning through Multilayer Fusion and Latent Dirichlet Allocation,” Frontiers in Robotics and AI 12, 2025, pp. 1579990.
[109] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” in China national conference on Chinese computational linguistics, Cham: Springer International Publishing, 2021, pp. 471-484.
[110] Y. Wang et al., “Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks,” in Proc. EMNLP 2022, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5085–5109.

[111] S. H. Bach et al., “PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts,” arXiv:2202.01279, Mar. 29, 2022.
[112] C. H. Song, H. J. Han, and Y. Avrithis, “All the attention you need: Global–local, spatial–channel attention for image retrieval,” in Proc. IEEE/CVF WACV 2022, Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 439–448.
[113] M. Röder, A. Both, and A. Hinneburg, “Exploring the space of topic coherence measures,” in Proc. WSDM 2015, Shanghai, China: ACM, Feb. 2015, pp. 399–408.
[114] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” arXiv:1607.06450, 2016.
[115] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance Normalization: The Missing Ingredient for Fast Stylization,” arXiv:1607.08022, Nov. 6, 2017.
[116] R. Socher et al., “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank,” in Proc. EMNLP 2013, Seattle, WA, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.
[117] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” in Proc. ACL 2005, Ann Arbor, MI, USA: Association for Computational Linguistics, 2005, pp. 115–124.
[118] X. Zhang, J. Zhao, and Y. LeCun, “Character-level Convolutional Networks for Text Classification,” Advances in neural information processing systems 28, 2015.
[119] S. Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?,” arXiv:2202.12837, Oct. 20, 2022.
[120] R. Xie, Y. Liu, S. Zhang, R. Wang, F. Xia, and L. Lin, “Personalized Approximate Pareto-Efficient Recommendation,” in Proc. The Web Conf. 2021, Ljubljana, Slovenia: ACM, Apr. 2021, pp. 3839–3849.
[121] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, “SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference,” arXiv:1808.05326, Aug. 15, 2018.