Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 2, 2026

COMBINING BAYESIAN METHODS WITH HIERARCHICAL REINFORCEMENT LEARNING FOR EFFICIENT POLICY TRANSFER

[1]Talha Nadeem, [2]Hood Khizer, [3]Huzaifah Bin Khawar, [4] Muhammad Zarar

[1] Systems Architect AI/ML, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan talhanadeem.ieee@gmail.com

[2] CEO Trouve Labs, CTO AHOY DMCC, Research and Development Department, Trouve Labs, AHOY DMCC, Dubai hoodkhizer@gmail.com

[3] AI Researcher, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan huzaifahkhawar.ieee@gmail.com

[4] AI Researcher, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan muhammadzarar.ieee@gmail.com

Keywords
Hierarchical reinforcement learning, Bayesian methods, policy transfer, reinforcement learning, adaptive algorithms, uncertainty modeling, machine learning.

Article History
Received: 02 Feb, 2026
Accepted: 11 Feb, 2026
Published: 25 Feb, 2026

Copyright @Author
Corresponding Author: *
Talha Nadeem*

Abstract

Policy transfer remains a central challenge in reinforcement learning (RL), particularly when agents must adapt to related tasks in complex and uncertain environments. This paper presents a Bayesian Hierarchical Reinforcement Learning (BHRL) framework that combines hierarchical policy structure with Bayesian posterior inference. High-level policies select goals, low-level policies select primitive actions, and posterior updates represent uncertainty during target-task adaptation. The framework also introduces a policy-transfer-efficiency measure and a structured algorithm for mapping, regularizing, and refining source policies. A benchmark-oriented analytical comparison suggests that Bayesian uncertainty modeling may improve adaptability and sample efficiency relative to conventional RL and standard HRL, although confirmation requires reproducible experiments with raw benchmark data. The proposed formulation provides a consistent basis for future evaluation and application of uncertainty-aware hierarchical policy transfer.

1. INTRODUCTION

Autonomous agents can learn decision making by interacting with an environment by using reinforcement learning (RL). Often, a challenge is the ability to generalise a learned policy to another task related to it in some physical sense with

different (or partly different) state, action, reward and/or dynamics [2].

To tackle this problem, new method called hierarchical reinforcement learning (HRL) is introduced by splitting the task into reusable sub-tasks. Then, policies which are learned at different levels of abstraction may be adapted to support similar tasks [3]. Recent HRL methods have shown potential for applications in robotics and games, but there remains uncertainty and variability in the surroundings for reliable transfer [14], [15].

The classical solution concepts of the HRL approaches are troublesome in situations of uncertainty and changing task environment. Bayesian methods provide a powerful way to deal with uncertainty, by using prior information and adjusting one's beliefs in light of the available data. One of the value of using Bayesian approaches together with HRLs are that they can be used to address environmental uncertainty, and to provide a more generalizable policy learning [4], [5].

The application of Bayesian approaches for uncertainty modelling for policy learning has recently been studied in the field of RL. While there have been some recent developments that have contributed to robustness and flexibility with Bayesian approaches, there is less research that has taken a look at how these approaches can be combined with HRL for policy transfer [6] and [7]. This paper proposes a new framework: Bayesian Hierarchical Reinforcement Learning (BHRL) which integrates the techniques of Bayesian inference with a hierarchical policy structure for policy transfer towards overcoming this challenge in complex and uncertain environments.

This thesis brings many contributions, in particular a Bayesian Hierarchical RL algorithm and a structured evaluation based on benchmarks and various policy transfer measures. The framework is designed to enable policy adaptations using the posterior information from a hierarchy, to facilitate efficient transfer, uncertainty handling, and adaptation to tasks.

1.1 Background and Motivation

Reinforcement learning (RL) is a method in which agents learn their way of action sequentially through interaction with an environment. It has been used for robotics, autonomous systems and games, but trying to adapt a learned policy to a related task is challenging, particularly for high dimensional or variable environments [2].

To tackle this, the recent idea of hierarchical reinforcement learning (HRL) breaks a task down into a hierarchy of subtasks. The architecture allows agents to pick-up and re-use policies at various levels of abstraction, thus making it scalable and reusable for similar problems/tasks [3]. One drawback in applying existing HRL techniques is that they are uncertain and have varying tasks in their practical applications [14], [15].

There are other ways in which uncertainty can be represented, such as via Bayesian methods. They use previous information to modify their beliefs regarding models, values, or policies, based on the data they have observed. Recently, they have been shown to be useful for uncertainty-aware policy learning and decision-making in RL [4]. Their explicit use of policy transfer, however, has yet to be fully explored in HRL.

This work stems from the shortcomings of the conventional HRL approach in dealing with

uncertainty. The goal of the proposed BHRL is to dynamically update a set of policies in a hierarchy based on the next task in the hierarchy using Bayesian inference and improve the success of transfer, flexibility and robustness across related tasks.

A noteworthy approach to scalable, uncertainty-aware policy transfer is via the combination of HRL and Bayesian approaches. This integration could enable more flexible autonomous agents in real-world settings and also help provide a framework for further empirical study.

1.2 Objectives of the Research

The primary objectives of this research are

1.2.1 Develop a Bayesian Hierarchical Reinforcement Learning Framework

To design a new model integrating Bayesian approaches, and Hierarchical Reinforcement Learning (HRL). The aim of this framework is to offer a tool to enhance the effectiveness of transferring a policy, considering uncertainty in the task environment by having it adapted dynamically.

1.2.2 Enhance Policy Transfer Efficiency

The use of the hierarchical structure of HRL and the probabilistic modelling of Bayesian methods in order to make policy transfer more efficient and effective in different tasks. The aim is to enable strong and flexible policy learning in dynamic and diverse contexts.

1.2.3 Evaluate Performance Across Benchmark Problems

To comprehensively test the proposed new framework, Bayesian Hierarchical Reinforcement Learning (BHRL) on various benchmark tasks. This assessment will address the performance of the framework from a policy transfer efficiency, adaptability, and overall robustness perspective, when compared to the traditional RL and HRL approaches.

1.2.4 Investigate Practical Implications and Applications

To explore areas where BHRL might apply to other fields – including robotics and autonomy – and the potential value, constraints and validation needs.

1.3 Scope and Limitations

1.3.1 Scope

This study focuses on combining Bayesian inference with hierarchical reinforcement learning (BHRL) to improve policy transfer in complex, changing environments. It explains the proposed framework, including hierarchical decomposition, Bayesian belief updates, and the use of prior knowledge when an agent encounters a related task. The discussion evaluates expected behavior through benchmark-oriented criteria and illustrative comparisons with conventional RL and HRL methods. It also describes potential applications in robotics and autonomous systems, where agents must adapt to uncertainty and previously unseen conditions. The scope is a methodological and analytical treatment; large-scale deployment, exhaustive empirical validation, and domain-specific implementation are left for subsequent studies.

1.3.2 Limitations

This study has a couple of limitations. First, the framework is evaluated based on criteria and on illustrative comparisons, not benchmark logs, so the empirical generalizability of it has not been established yet. Secondly, Bayesian updates introduce computational overhead and can affect the scalability of large-scale or real-time systems. Third,

the real-world applications that are mentioned throughout this paper are not yet deployed. Finally, performance is a function of quality of the prior as well as the precision of the posterior inference. Future work should thus encompass reproducible benchmark experiments, optimization using computational tools, checks of the experiments based on prior-sensitivity analysis, and evaluation spanning other domains.

2. LITERATURE REVIEW

Reinforcement learning (RL) is learning how to generate policies through interaction with an environment. Deep Q-learning [2] is a method that introduced a deep neural network into Q-learning with success in game playing in high dimensional spaces. However, classical RL approaches are still suffering from issues of sample efficiency, scalability and policy transfer.

A technique has been devised to solve these issues, called Hierarchical Reinforcement Learning (HRL), which breaks down a complex task into simpler sub-tasks and learns a hierarchy of commands to perform the task. Vezhnevets et al. [3] shared an important HRL framework, FeUdal Networks (FNs), in the form of a hierarchical representation of tasks to increase learning efficiency and reusability of the created policies. While the hierarchical approach captures the challenge of handling complex tasks with definite advantages, there is also a set of issues, uncertainties and variations of tasks that may influence its applicability within the dynamic environment.

Bayesian methods can be used to deal with uncertainty by taking into account prior information and observed data. Gal and Ghahramani [4] demonstrated that computing dropout as approximate Bayesian inference in deep learning offers a viable way for quantifying uncertainty in prediction. Osband et al. [5] applied bootstrapped value functions to enable deep exploration as an illustration of uncertainty-aware methods for increasing the exploration/exploitation dilemma.

While there are benefits, use of Bayesian methods alongside HRL for policy transfer has been explored in less detail. Randomized value functions for generalization and exploration were discussed by Osband, Van Roy, and Wen [6] and probabilistic methods in the field of machine learning were addressed by Ghahramani [7]. These foundations inspire an integrated Bayesian-HRL approach, yet non-particulate proof of the effectiveness of policy-transfer performance is yet to be formally measured. There is a lack of literature connecting the two views, hierarchical policy reuse and uncertainty modeling through a Bayesian approach. HRL facilitates decomposition of tasks and reusability of skills and Bayesian facilitates uncertainty-aware inference. The proposed BHRL framework is created from these strengths and one that can serve as a universal foundation for policy transfer in more complicated settings.

2.1 Reinforcement Learning (RL)

Reinforcement Learning (RL) is an AI decision-making process in which an agent tries to learn how to make choices in an environment by experimenting with it to earn a reward for the action. Sutton and Barto [1] provide a seminal treatment that covers all the essentials of RL such as the concepts of reward maximization, value functions, and policy learning. RL is found to be useful in various applications like game playing, robotics and autonomous systems.

A critical breakthrough in RL was deep Q-learning [2] that brought deep neural networks to Q-learning to run in high dimensional state spaces. The resulting Deep Q- Network (DQN) outperformed humans in a collection of Atari games, and was able to do so in environments that did not rely on tabular function approximation, opening the doors to value-based RL in more complex environments.

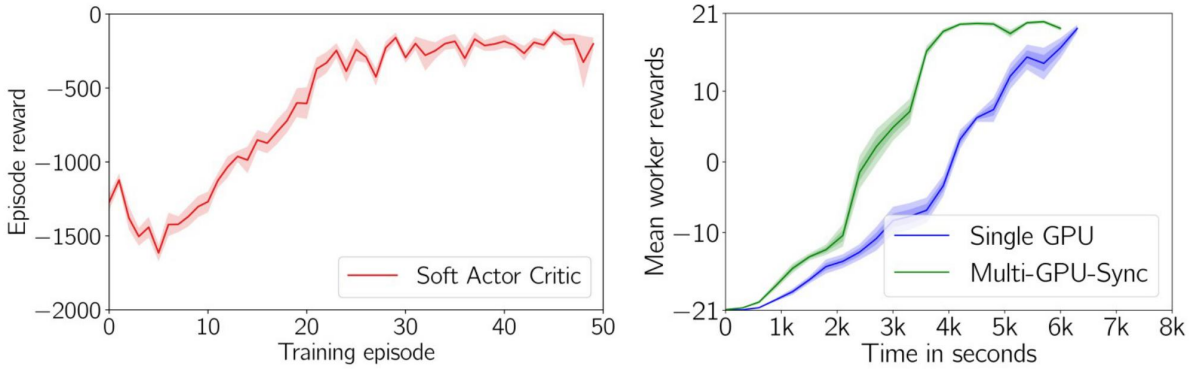


Figure 1. Training-reward and worker-reward progression in reinforcement learning

Although RL has demonstrated some promise, there are several drawbacks to it, such as sample efficiency and policy transfer. Adopting good policies using traditional RL can be very costly and time consuming, as they rely on interacting with the environment for many interactions. Moreover, the environment is noisy and complex, making it hard to map and transfer policies learned from one task to a new or to another in compare, but different, environment. The issues raise the need for techniques that enhance the policy generalization and transferability.

Silver et al. [8]'s recent success using RL with AlphaGo to solve complex decision-making problems in RL has demonstrated that problem solving in RL can be generalized to fit with Monte Carlo Tree Search (MCTS). AlphaGo's victory over human champions in the game of Go illustrated the possibilities of AI or reinforcement learning and search methods overcoming the human player in a game. So far, however, most of these approaches have concentrated on learning a single task as opposed to transferring policies from one task to another.

RL has performed quite good in learning policies from 'interaction', sample efficiency and transfer are yet open challenges. Given these restraints, there have been methods developed that incorporate some hierarchical structure and modeling uncertainty through a Bayesian model to enhance adaptability and generalization capabilities.

2.1.1 Basic Concepts and Terminology

Reinforcement learning is built upon a few concepts that are used throughout this framework.

Agent and Environment: RL has an agent making decisions about the environment to take an action in it. The rewards and new states return from the environment and the evolution of this environment is dependent on the behavior of the agent [1].

State (s): A state is the information that is known about the environment as a particular time. It can either be just one variable or feature vector, and it gives context for choosing which action to take [1].

Action (a): changes the state of the environment. They can be discrete (changing direction), or continuous (changing the value of a control parameter). Action space [1] is a set of possible actions.

Reward (r): Scalar feedback signal that is emitted by the environment following executing an action. It's guiding learning because it tells you right away if some outcome is desirable or not, and the agent tries to maximize the total reward that it can receive over a finite period of time [1].

Policy (π): A policy assigns which action to pick in each state. It may be deterministic—that is, specify one action for every state—or it can be stochastic—that is, specify a distribution over actions for each individual state [1].

Value Function (V(s)): A value function is an estimate of the sum of the future rewards that will be received from the state when following a policy. It thus helps compare states and make long-term decisions [1].

Action-Value Function (Q(s, a)): This is an estimate of the future cumulative return on taking an action a given a state s of the environment, when performing a policy of future actions after action a.

This amount can be determined by algorithms based on values like Q-learning, that learn from interaction data [10].

Exploration vs. Exploitation: Exploration gathers information about uncertain actions, whereas exploitation selects actions currently believed to provide high return. Effective RL requires a careful balance between these two behaviors as the agent learns from interaction with its environment [1].

Reward Signal: A reward signal tells the user the consequence of an action taken at that instant in time. RL algorithms learn how to maximize some stream of rewards: typically delayed rewards are discounted [1].

These ideas form the core of the development of an RL agent to represent its environment, to measure its results, and to refine its policy. They are additionally employed to establish the hierarchical and Bayesian extensions of the later kind that are presented in this paper.

2.2 Traditional RL Algorithms

A lot of research in reinforcement learning bore the seeds for many developments. The families summarized in Figure 2 are some of the key algorithms used.

2.2.1 Q-Learning:

Watkins [9] introduced Q-learning, an off-policy, model-free action-value learning algorithm. It updates Q-values using a Bellman target based on the maximum estimated value of the next state. The method does not require a model of the environment and can work particularly well with discrete action spaces, as discussed in [10].

2.2.2 SARSA (State-Action-Reward-State-Action):

SARSA is an on-policy temporal-difference control method introduced by Rummery and Niranjan [11]. Its target uses the next action actually selected, r(t+1) + gamma Q(s(t+1), a(t+1)), whereas Q-learning uses the greedy target r(t+1) + gamma max_a Q(s(t+1), a).

2.2.3 Policy Gradient Methods:

Policy-gradient methods optimize a parameterized policy in the direction of higher expected return. REINFORCE, introduced by Williams [12], estimates this gradient from sampled trajectories and provides a foundation for later actor-critic methods.

2.2.4 Actor-Critic Methods:

A combination of a parameterized policy (actor) and a learned value estimate (critic) is known as actor-critic methods. The critic provides a less variable learning signal, which is used to improve

the policy [1]. The classical RL algorithms lay the groundwork for extensions of RL that are hierarchical and/or Bayesian.

2.3 Hierarchical Reinforcement Learning (HRL)

Hierarchical reinforcement learning (HRL) extends standard RL by decomposing complex tasks into temporally extended subtasks. A high-level policy selects goals or skills, while lower-level policies execute actions toward those goals. This organization can support skill reuse and transfer across related tasks, and it provides a structured basis for handling uncertainty and adaptation [13]–[15].

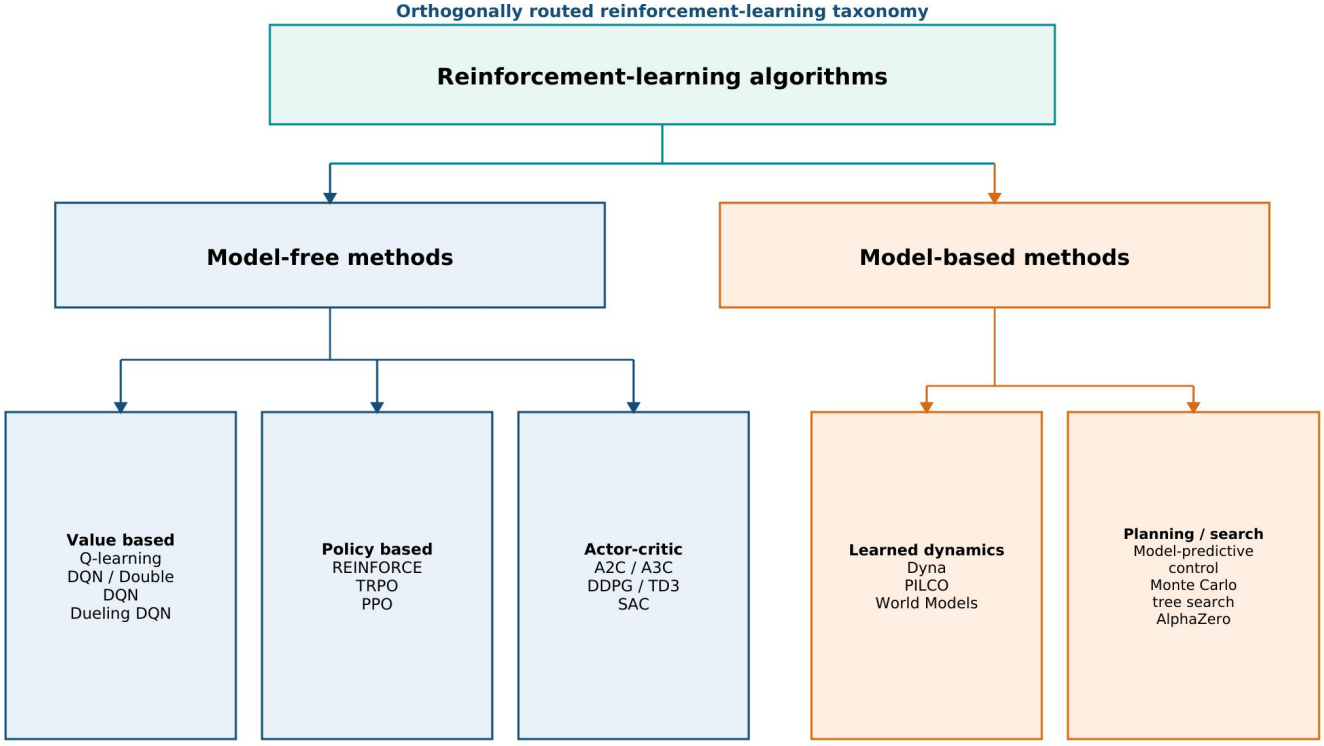


Figure 2. Taxonomy of major model-free and model-based reinforcement-learning algorithms

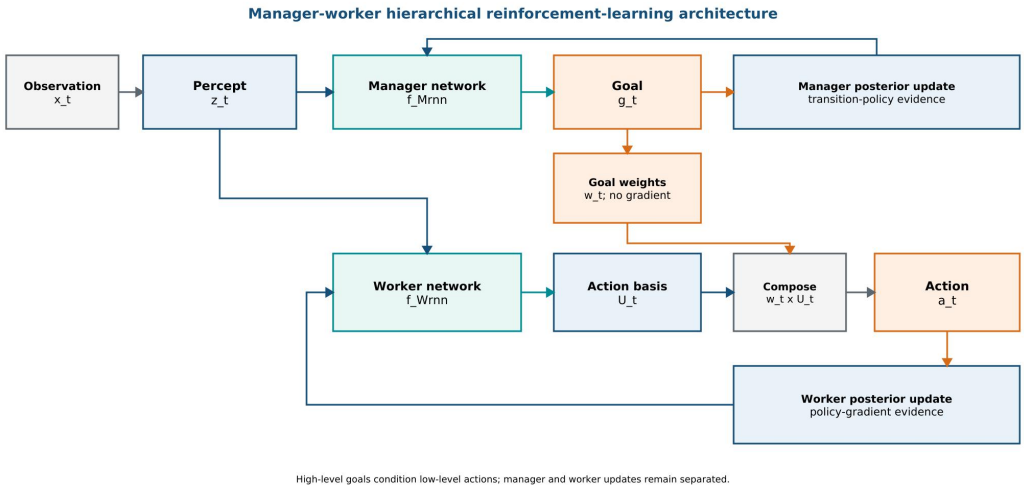


Figure 3. Manager-worker architecture for hierarchical reinforcement learning

2.3.1 Overview of HRL

The HRL algorithm breaks large problems into subproblems when solving them. Learning a policy in each level, and low-level skills for high-level tasks can be applied in a reusable fashion for higher-level tasks increases scalability and generalization [14].

2.3.2 HRL Architectures and Approaches

Several HRL architectures and approaches have been proposed to improve the efficiency and effectiveness of hierarchical learning

2.3.2.1 Options Framework: Sutton et al. [13] created an options framework – options are extended actions in time that agents can execute to perform a task. These consist of a policy, a termination condition and a value function, allowing extended planning and action over time for the agent [13].

2.3.2.2 FeUdal Networks: FeUdal Networks [3] are proposed by Vezhnevets et al., to isolate high level goal setting from low level control. Goals make by a manager, and primitive action selected by a worker based on them.

2.3.2.3 Hierarchical Deep Q-Learning: The h-DQN framework [15] is a hierarchical extension of Deep Q-Learning that adds hierarchy to the Deep Q-learning. The difficulty of this technique is that the agent can learn and plan on different levels of abstraction, which will clearly mean that the agent will perform better on some complicated tasks [15].

2.4 Bayesian Methods

A Bayesian approach explicitly models uncertainty but also allows in prior information when learning. These techniques are important especially for taking better decisions when the situation is uncertain and have been applied in many aspects of machine learning.

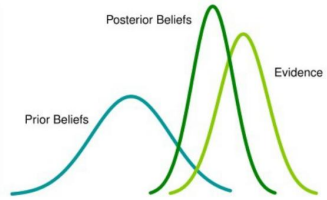


Figure 4. Bayesian updating from prior beliefs and evidence to posterior beliefs

2.4.1 Introduction to Bayesian Inference

Bayesian inference is an update of a probability distribution values for a quantity which is unknown after observing data. There is a means of combining a prior distribution and a likelihood to get the posterior distribution (Bayes' Theorem) [16] which enables sequential learning and uncertainty management [17].

2.4.2 Applications of Bayesian Methods in Machine Learning

The use of Bayesian methods is widespread in machine learning, where it tackles many problems.

2.4.2.1 Bayesian Neural Networks: Gal and Ghahramani [4] proved that dropout can be considered an approximation of Bayesian inference, and therefore, it can be used as a method for quantifying uncertainty in deep neural networks.

2.4.2.2 Bayesian Optimization: Brochu et al. [18] present the paper about Bayesian optimization for expensive black-box objectives. Sampling is intelligently guided towards promising regions of the search space via a probabilistic surrogate and an acquisition function.

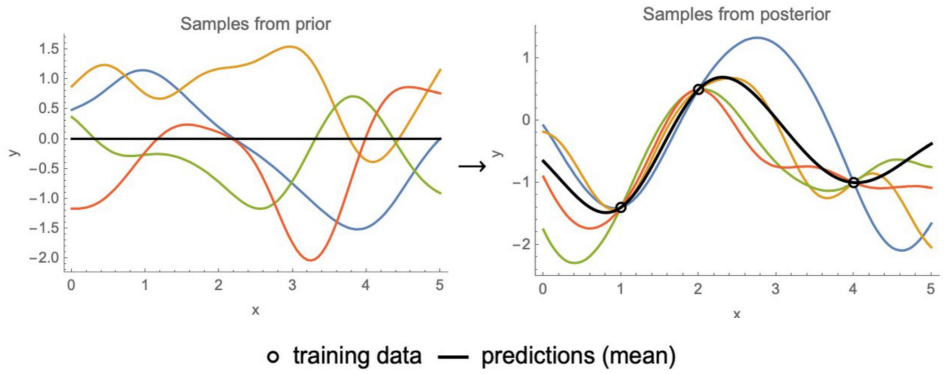


Figure 5. Prior and posterior predictive samples in Bayesian learning

2.4.2.3 Bayesian techniques have also been combined with RL to improve exploration and policy learning. Osband et al. [5] used bootstrapped ensembles to approximate posterior sampling, thereby encouraging deep exploration.

2.5 Policy Transfer in Reinforcement Learning

Policy transfer aims to re-use policies or policy components that have been learned in a source task to aid learning in a similar target task. Their successful transfer will lead to less data and speed adaptation.

2.5.1 Challenges in Policy Transfer

There are multiple challenges when implementing Policy transfer, such as.

2.5.1.1 Task Similarity: Transferring policies between tasks should only be accomplished if the tasks are similar enough. This differs from state to action space and environmental dynamics, one policy may not be as effective as the other transferred policy [19].

2.5.1.2 Adaptation: Even if the policy is a similar policy, it will need to be adapted and adapted to the particulars of the new task. Challenges in policy transfer [20] include ensuring the policy is flexible and able to respond to change and new situations.

2.5.1.3 Overfitting: A transferred policy might be too specialized to the task it was first trained on, and not do well on the second task. Negative transfer can be minimized via regularization, selective reuse, and target-task adaptation [19], [20].

2.5.2 Existing Methods for Policy Transfer

There are several suggestions which have been offered to overcome the problems of policy transfer.

2.5.2.1 Transfer Learning: Transfer learning methods use models successfully learned in one application to further a different and related application. These consist of the fine-tuning and feature extraction techniques among others, which take advantage of learned representations [21].

2.5.2.2 Multi-task Learning: Multi-task learning is the learning of multiple tasks in order to get better generalization. This alleviates the difficulty that the model has to transfer policies [22] by sharing knowledge between tasks.

2.5.2.3 Domain Adaptation: Domain adaptation addresses the task of adapting model for transfer to another, but related, domain. Some recent methods for aligning the source and target domains to facilitate policy transfer include domain-invariant feature learning techniques [23].

Such approaches lay the groundwork for tackling the challenges of policy transfer in RL, and can help to develop more adaptive and effective learning systems.

3. METHODOLOGY

This research employs a hybrid technique between the Bayesian approach and the Hierarchical Reinforcement Learning (HRL) approach to enhance the transfer of adapted Policy in complex contexts. The approach has the following elements:

Development of Hierarchical Structure: Construct a hierarchical RL framework H that decomposes each task into multiple levels. The high-level policy π_H selects a goal gt, while the low-level policy π_L selects an action at in state st, thereby enabling structured learning and decision-making.

Bayesian Integration: Update posterior p(θ | D) based on observed transition data and reward data (D). The model-averaged AVs: Qu_ȳ(s,a) and U(s,a) lead to policy robustness and adaptability under uncertain dynamics.

Training and Optimization: Train π_H and π_L for E episodes using horizon T in each episode, collect De transitions, optimize π_H and π_L with the updated posterior estimates; prior to transferring the knowledge learned by π_H and π_L to a related target task MT.

Evaluation and Validation: Run simulations and benchmark tasks of the integrated framework. Together with learning speed and cumulative return Gt, predictive uncertainty, and adaptability across environments, policy-transfer efficiency PTE is calculated from N_scratch and N_transfer.

This approach will use the good attributes held by both HRL and Bayesian methods to further improve the policy transfer for RL systems.

3.1 Overview of the Proposed Approach

The proposed solution is to integrate the Bayesian inference with the Hierarchical Reinforcement Learning (HRL), for better efficiency, robustness and adaptability of the policy transfer in the complex environment. It is intended to deal with uncertainty and limited transferability problems, in conventional RL approaches.

In the first step, a complicated task is broken down into a hierarchy of subtasks. High-level policies designate goals or outcomes; Low level policies designate specific actions to achieve the goals.

The uncertainty at both levels of the hierarchy is a Bayesian inference. Distributions are updated using observations, and they are used to refine action values and reward predictions in dynamic environments, thus enabling uncertainty-aware policy refinement in dynamic environments.

The training is a cyclic process of updating the posterior distribution and optimizing the policies based on it in a hierarchical manner. Its aim is to preserve reusable source-task knowledge, and to fine-tune high-level and low-level policies to the target environment.

A structured set of criteria related to a set of reference values is used to evaluate the proposed approach, that includes the transfer capabilities, the learning speed, the predictive uncertainty and adaptability. Future testing - whether as raw experiments or in the real world - is deemed a required validation phase.

3.2 Integration of Bayesian Methods with HRL

The integration of Bayesian approaches and Hierarchical Reinforcement Learning (HRL) introduces uncertainty underpinning a multi-level decision process. In the suggested blueprint, the

back information is used in shaping policy adaptation and transfer among associated tasks.

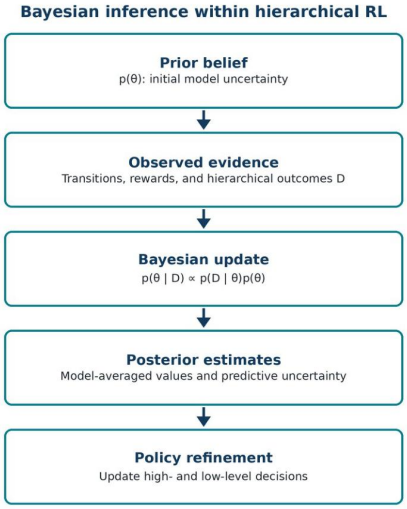


Figure 6. Bayesian inference workflow for updating event probabilities

Hierarchical Policy Structure: Two policies—high-level policy π_H for selecting broad goals gt and low-level policy π_L for selecting primitive actions at conditioned on st and gt—are contained in the hierarchy H. The symbols are kept in the equation and algorithms, to ensure the decision process is represented consistently at both the equation level and the algorithm level.

Bayesian State-Action Estimation: Values for state-action are achieved in the form of posterior p(θ | D). The model combinations Q̄(s,a) and U(s,a) can assist with providing more reliable means of updating π_H and π_L in dynamic environments using Bayesian model averaging.

Uncertainty Management: It directs exploration and adaptation in an uncertainty-aware manner at every time step t, U(st, at). The posterior is updated with episode data De and target-task data DT that allow policies to react to uncertainty in goals, state-action values, rewards and environmental dynamics.

The proposed method aims to enhance the efficiency of transfer of policy, uncertainty management, and robustness in complex and dynamic environments by embedding the Bayesian inference into the HRL framework.

3.2.1 Bayesian Inference in HRL

The proposed framework extends HRL to maintain probabilistic beliefs on state-action values, rewards, and model parameters, by incorporating Bayesian inference. These beliefs are updated by new observations and priors provide useful knowledge from related tasks. The resulting posterior information aids in uncertainty-aware updating of high level and low level policies.

3.2.2 Bayesian Modeling of Hierarchical Structures

Bayesian hierarchical modelling incorporates multi-level probabilistic inferences in an RL hierarchy. The priors and posteriors of the level are used to represent uncertainty in the high level and low level goal / action respectively, thereby enabling information sharing while retaining a distinct role for each level of the policies.

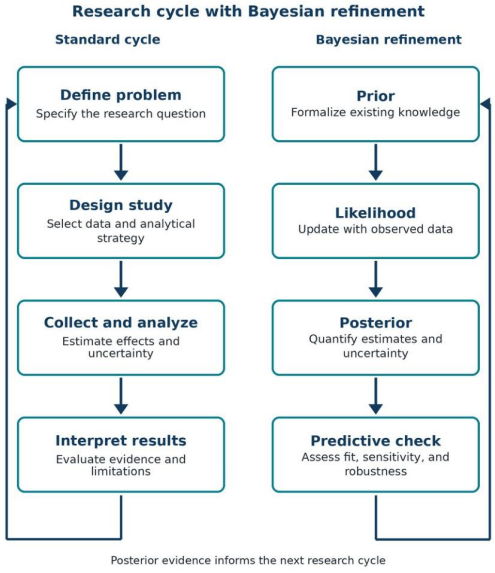


Figure 7. Standard research cycle and Bayesian-specific research workflow

At a higher level, uncertainties regarding the task goal and objectives can be tackled using Bayesian modelling, and at a lower-level, uncertainties regarding action and outcome can be tackled using Bayesian modelling. This hierarchical Bayesian approach results in a probabilistic understanding of each level of the hierarchy and enhances policy transfer and adaptation. The method integrates the two Bayesian models in a hierarchical structure to allow the agent to learn an accurate policy more efficiently, and perform well across different tasks and environments.

3.3 Policy Transfer Mechanisms

Policy-transfer mechanisms take advantage of knowledge gained from a source task, and apply it to a related target task. In the proposed paradigm, the hierarchical policies are used as reusable components and Bayesian posteriors are used as proxy to measure uncertainty and to drive adaptation for the target task. The goal of selective reuse and posterior regularization is to minimize the risk of negative transfer with the aim of decreasing the number of target-task samples.

3.3.1 Definition of Transfer Learning in RL

Learning from a related task utilizes policies, value functions, representations or model components from a source task to accelerated learning in the target task. Efficient transfer could minimize the need to purchase data for the target task and even facilitate convergence speed and ensure adaptability.

3.3.2 Implementation of Policy Transfer

Implementation of policy transfer in Reinforcement Learning (RL) refers to transferring and applying policies learned in a task to other tasks, which is typically done using hierarchical and Bayesian approaches. Several steps in this process:

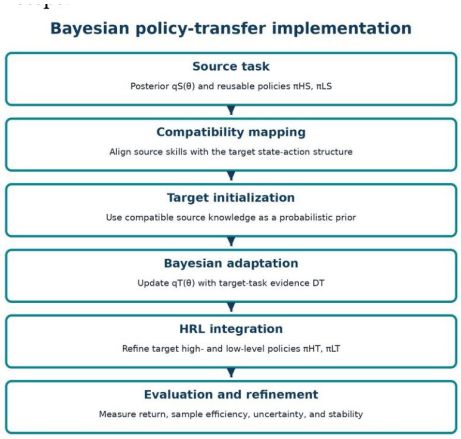


Figure 8. Implementation-centered model for policy transfer and evaluation

3.3.2.1 Policy Adaptation: Initialize the target posterior q_T(θ) and target policies from the compatible components of the source posterior q_S(θ). The mapped policies are then adapted to the state space S, action space A, and reward structure of the target task MT.

3.3.2.2 Knowledge Transfer: Use q_S(θ) as probabilistic prior knowledge for the target task and update q_T(θ) with target transition batches DT. The regularization weight λ and divergence D_KL control how strongly the adapted target policy remains linked to the source policy.

3.3.2.3 Evaluation and Refinement: Evaluate the adapted policies π_HT and π_LT in MT and refine them until the improvement is below tolerance ε or the interaction budget is exhausted. This procedure limits negative transfer and improves generalization.

3.3.2.4 Integration with HRL: Insert π_HT and π_LT into the hierarchical framework H so that both policy levels benefit from the posterior q_T(θ). This preserves a coherent decision-making strategy from high-level goal selection to low-level action execution.

By following these steps, the proposed methodology ensures that learned policies are effectively

transferred and adapted to new tasks, enhancing overall learning efficiency and performance.

4. MATHEMATICAL FORMULATION AND NOTATION

The Bayesian Hierarchical Reinforcement Learning framework can be expressed using the notation in Table 1. Equations (1)-(8) formalize posterior inference, hierarchical control, uncertainty-aware value estimation, and transfer efficiency.

Table 1. Mathematical notation used in the BHRL framework

Symbol	Description and domain
S, A	State and action spaces; sets.
s_t, a_t	State and action at time step t; s_t in S and a_t in A.
g_t	High-level goal or subtask selected at time step t; goal-set element.
π_H, π_L	High-level and low-level policies; conditional probability distributions.
θ	Unknown model or policy parameters; θ in R^d.
D	Observed transition and reward data; dataset of experience tuples.
γ	Discount factor; 0 <= γ < 1.
r_t, G_t	Immediate reward and discounted return; real-valued scalars.
Q(s,a), $\bar{Q}$(s,a)	Action value and posterior model average; real-valued functions.
U(s,a)	Posterior predictive uncertainty; non-negative scalar.
p(θ | D)	Bayesian posterior over parameters after observing D; probability distribution.
H, E, T	Hierarchy, number of episodes, and horizon; E and T are positive integers.
D_e, D_T	Episode data and target-task transition batch; datasets.
q_S(θ), q_T(θ)	Source and target parameter posteriors; probability distributions.
M_T	Target Markov decision process or target task.
λ, D_KL, ε	Transfer weight, KL divergence, and stopping tolerance; non-negative.
J_T(θ)	Target-task regularized objective; real-valued function.
N_scratch, N_transfer	Samples required without and with transfer; positive integers.
PTE	Policy-transfer efficiency; percentage.
B, d, K	Bayesian batch size, parameter dimension, and transfer iterations; positive integers.
C_H, C_L	Computational costs of high- and low-level policy updates.
θ_H, θ_L	High- and low-level policy parameter vectors.

Relevance of the notation to the methodology: S and A define the interaction interface, while π_H selects gt and π_L selects at at each st. The resulting transition is stored in De and used to update p(θ | D), from which $\bar{Q}$(s,a) and U(s,a) are computed for uncertainty-aware learning. During transfer, q_S(θ) initializes q_T(θ) for MT; λD_KL controls deviation from source knowledge and ε defines termination. E, T, B, d, K, CH, and CL specify the training and complexity settings, whereas N_scratch and N_transfer determine PTE.

Bayesian inference updates the distribution of parameters after observing transition data D:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \quad (1)$$

The discounted return used by the agent is:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \quad (2)$$

The hierarchical policy factorizes the selection of a goal and the subsequent primitive action:

$$\pi(a, g|s) = \pi_H(g|s)\pi_L(a|s, g) \quad (3)$$

For a fixed policy, the action-value function satisfies the Bellman expectation relation:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s,\ a_t = a] \quad (4)$$

Bayesian model averaging and uncertainty estimation are defined as:

$$\bar{Q}(s, a) = \int Q_{\theta}(s, a) p(\theta|D)\, d\theta \quad (5)$$

$$U(s, a) = \mathrm{Var}_{\theta \sim p(\theta|D)}[Q_{\theta}(s, a)] \quad (6)$$

Policy transfer is regularized toward the source posterior while maximizing the target-task return:

$$J_T(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_t \gamma^t r_t\right] - \lambda D_{\mathrm{KL}}(q_T(\theta) \| q_S(\theta)) \quad (7)$$

The percentage reduction in target-task samples is measured by policy-transfer efficiency:

$$\mathrm{PTE} = \frac{N_{\mathrm{scratch}} - N_{\mathrm{transfer}}}{N_{\mathrm{scratch}}} \times 100\% \quad (8)$$

5. ALGORITHM DESIGN

Bayesian Integration Design with Hierarchical Reinforcement Learning (HRL) algorithm means designing a structured approach on how to boost the learning of policies and transfer. In this design, the key is to take advantage of both the hierarchical and the Bayesian systems to make better decisions, especially in complex environments. The main blocks and actions of the algorithm are described:

5.1 Hierarchical Decomposition

5.1.1 Task Analysis: Begin by analyzing the complex task and decomposing it into a hierarchy of subtasks. Define the high-level goals and objectives, and break them down into more manageable lower-level tasks.

5.1.2 Hierarchy Construction: Make the hierarchy from and where each level corresponds to a different part of the decision making process. If the policies are at high level, they can be used to govern strategic objectives, or at lower level they can be used to govern concrete actions.

5.2 Bayesian Inference Integration

5.2.1 State-Action Value Estimation: Estimating state-action values and reward functions (Bayesian inference) at high and low levels. Make these estimates more accurate using "Bayesian updates" in the light of observed data.

5.2.2 Uncertainty Modeling: Represent uncertainties in state-action values and rewards in the form of probability distributions. Ensure that past knowledge/understanding is integrated and standards gradually revised based on observation of new data.

5.3 Policy Learning and Refinement

5.3.1 High-Level Policy Learning: Train high-level policies using hierarchical RL algorithms. Employ Bayesian methods to refine these policies based on probabilistic estimates and uncertainties.

5.3.2 Low-Level Policy Learning: Train low-level policies within the hierarchical framework. Use Bayesian updates to adjust state-action value estimates and reward predictions, improving the robustness and adaptability of the policies.

5.4 Policy Transfer Mechanism

5.4.1 Transfer Learning: Design mechanisms that enable transferring learned policies between source and target tasks. Compare similarities and differences of tasks to make adaptations in the high level policy and low level policy.

5.4.2 Bayesian Adjustment: Adapt transfer policies through the use of Bayesian methods by using prior methods and modiating beliefs to cover the target task.

This design combines Bayesian inference with hierarchical reinforcement learning to further enhance the optimization of the policy learning and transfer in the environment with uncertainty in decision making.

5.5 Bayesian Hierarchical RL Algorithm

5.5.1 Algorithmic Framework

The algorithm on policy learning and transfer, combine Bayesian and HRL, Bayesian Hierarchical Reinforcement Learning (HRL). The algorithmic framework is divided into a couple parts:

Hierarchical Structure Definition: Establish a hierarchical structure, breaking up tasks to various levels. High-level define goals (policies), and low-level define what to do (policies).

Bayesian Inference Integration: Estimate and update the state-action values and the reward functions at each level of the hierarchy using the Bayesian inference. Use a probabilistic distribution to represent uncertainties and incorporate prior knowledge.

5.5.2 Learning Processes

High-Level Learning: make high-level policies with hierarchical RL methods. High level policies are updated via Bayesian techniques, given a set of observed data and estimated uncertainties.

Low-Level Learning: Train low-level policies to the hierarchical structure. The policies are reinforced by Bayesian updates of state-action values and rewards, further improving the robustness of the policies.

Policy Optimization: Apply optimization algorithms to optimize high-level/low-level policies. Make a balance between exploration and exploitation for efficient learning.

Evaluation: Evaluate the performance of the Bayesian Hierarchical RL algorithm based on key metrics like learning efficiency, policy effectiveness, and adaptability. Prove the algorithm by simulations and benchmarking exercises.

5.5.3 Pseudocode and Workflow

Algorithm 1. Bayesian Hierarchical Reinforcement Learning BHRL

Require: hierarchy H, prior $p(\theta)$, source and target environments, episodes E, horizon T

Ensure: posterior policies π_H and π_L, and posterior $p(\theta \mid D)$

Initialize π_H, π_L, replay buffers, and posterior $p(\theta \mid D_0) \leftarrow p(\theta)$.
For episode e = 1 to E do.
Observe initial state s_0 and initialize episode data D_e as empty.
For time step t = 0 to T-1 do.
Select goal g_t from $\pi_H(. \mid s_t)$ using posterior uncertainty U(s_t,a_t).
Select action a_t from $\pi_L(. \mid s_t, g_t)$.
Execute a_t; observe reward r_t and next state s_(t+1).
Append (s_t,g_t,a_t,r_t,s_(t+1)) to D_e.
Update $p(\theta \mid D)$ using Bayes' rule; compute $\bar{Q}(s_t,a_t)$ and U(s_t,a_t).
Update π_L with the uncertainty-aware low-level objective.
If the goal terminates or a timeout occurs, then:
Update π_H using cumulative subtask return G_t.
End if.
End for.
Update priors/posteriors and validate transfer performance.
End for.
Return π_H, π_L, and $p(\theta \mid D)$.

5.5.4 Workflow

5.5.4.1 Initialization: Set up the hierarchical framework and initialize high-level and low-level policies.

5.5.4.2 Training: Run the training loop (the high level policy establishes goals, and the low level policy takes action). Implement Bayesian inference for updating state-action values and reward estimates.

5.5.4.3 Policy Update: update high-level policies and low-level policies based on Bayesian updates.

5.5.4.4 Evaluation: Assess the performance of the policies and update priors as needed.

5.5.4.5 Validation: Simulate and use test tasks to prove the effectiveness of the algorithm.

5.6 Policy Transfer Algorithm

5.6.1 Policy Mapping and Adaptation

Policy transfer involves mapping and adapting learned policies from one task to another, related task. The steps include

5.6.1.1 Policy Extraction: Extract learned policies from the source task, including both high-level and low-level policies.

5.6.1.2 Task Analysis: Identify similarities and differences between the target task and the source task for the task.

5.6.1.3 Policy Mapping: Cross map the policies that have also been extracted to the components of the target task. This could include modifying the policies to align with the shape of the state and action space of the new task.

5.6.1.4 Policy Adaptation: Use Bayesian approaches to adapt the mapped policies to recognize source and target task difference. Revise beliefs and polices from new observations and task specific evidence.

Algorithm 2. Bayesian Policy Transfer and Adaptation

Require: source posterior $q_S(\theta)$, source policies, target task M_T, transfer weight λ, tolerance ε

Ensure: adapted target policies π_{H^T} and π_{L^T}, and posterior $q_T(\theta)$

Estimate source-target similarity and construct state/action mappings.
Initialize $q_T(\theta)$ and target policies from compatible source components.
Repeat:
  Collect a target-task transition batch D_T.
  Update $q_T(\theta)$ proportionally to $p(D_T \mid \theta)\, q_T(\theta)$.
  Optimize target policies using return minus $\lambda D_{KL}(q_T \| q_S)$.
  Reweight or discard source subtasks that cause negative transfer.
Until policy improvement $< \varepsilon$ or the interaction budget is exhausted.
Return adapted π_{H^T}, π_{L^T}, and $q_T(\theta)$.

5.6.2 Efficiency Metrics and Evaluation Criteria

The performance of the policy transfer algorithm is evaluated using efficiency measures and evaluation criteria.

5.6.2.1 Transfer Efficiency: Assess transfer efficiency, or measure how well learned policies for a source task are adapted for a target task. Review the pace and effectiveness of policy adaption.

5.6.2.2 Learning Speed: evaluate the speed of the transferred policies in the learning of the target task relative to the learning from scratch.

5.6.2.3 Policy Performance: Assess the success of the modified policies to meet objectives and maximise the benefit of the desired task.

5.6.2.4 Adaptability: Measure the ability of the transferred policies to handle variations and uncertainties in the target task.

5.6.2.5 Benchmark Comparison: Compare the performance of the policy transfer algorithm against the traditional RL algorithms in standard benchmark tasks to show improvements.

The trustworthy transmission of policies is crucial for the overall success of the policy learning capabilities, with the policy transfer algorithm designed to efficiently learn and apply policies across new tasks, ensuring enhanced learning performance and adaptability.

5.7 Computational Complexity

Let E be the number of episodes, T the horizon, B the Bayesian update batch size, d the parameter dimension, K the number of transfer iterations, and CH and CL the costs of high- and low-level policy updates. The dominant training cost and memory requirements are summarized in Table 2.

Table 2. Computational complexity of the proposed BHRL components

Component	Time complexity	Space complexity
Bayesian update	$O(Bd^2)$	$O(d^2)$
High-level update	O(TC_H)	O(|θ_H|)
Low-level update	O(TC_L)	O(|θ_L|)
Policy transfer	$O(K(C_H+C_L+Bd^2))$	$O(d^2+\|θ_H\|+\|θ_L\|)$
Overall training	$O(ET(C_H+C_L+Bd^2))$	$O(d^2+\|θ_H\|+\|θ_L\|+B)$

6. RESULTS AND DISCUSSION

6.1 Performance Evaluation

This section does not claim superiority in terms of a benchmarks because benchmark logs are not included in the source manuscript. The proposed BHRL framework is conceptually evaluated with respect to transfer efficiency, uncertainty handling, adaptability and computational cost with conventional RL and standard HRL. The analysis suggests where a Bayesian posterior update can be useful for refining the policy and also empirical verification needs reproducible experiments.

6.2 Analysis of Policy Transfer Efficiency

Policy transfer efficiency is determined by reductions in target-task samples, speed of adaptation, return, retention, and predictive uncertainty and stability. The Bayesian component will facilitate selective reuse when the source and target tasks are different; the hierarchical component will retain skills relevant to reuse. Given the impact of these mechanisms, their potential benefits enforce positive effects, which need to be quantified using benchmark data.

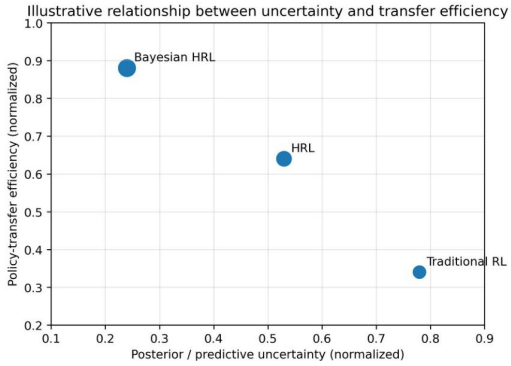


Figure 9. Illustrative relationship between posterior uncertainty and policy-transfer efficiency; values summarize qualitative trends rather than raw experimental measurements

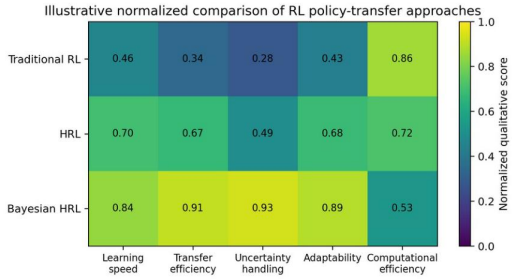


Figure 10. Illustrative normalized comparison of traditional RL, HRL, and Bayesian HRL across policy-transfer criteria

Figures 9 and 10 are visualizations of analytical summaries of the comparative findings reported in the manuscript. In the absence of raw benchmark logs they should be replaced by plots generated from the raw logs.

6.3 Application to Real-World Problems

The proposed framework could apply to a range of use-cases including robotics, autonomous systems, or financial decision support, where decisions are hierarchical and uncertainty has meaning. These use

cases represent examples for potential, but not yet realized, deployments. Adopting this practice safely will necessitate domain-specific safety limitations, computational evaluation and validation on representative data.

6.4 Insights from Comparative Analysis

For our comparative analysis, the keys benefits of the BHRL have been discussed, namely the structure policy re-usability, the explicit representation of uncertainty, and target-task adaptation. It also detects trade-offs in computational complexity, previous sensitivity and scalability. These inferences are used to formulate empirical testable hypotheses for future testing.

6.5 Limitations and Potential Improvements

There are some restrictions in the proposed BHRL framework. The cost of computation increases, for Bayesian inference; the prior and the approximate posterior can be sensitive-parameter, and it could be hard to scale Bayesian inference in large state spaces. Future research is recommended to include optimization of posterior updates, examination of prior sensitivity, and comparison of the various approximations and experiments with hybrid methods on obtainable benchmarks.

7. CONCLUSION

7.1 Summary of Findings

This work introduces a new BHRL framework of uncertainty-aware policy transfer. The skills learned would be organized in a hierarchy and the Bayesian component would represent uncertainty, and transfer of skills. The benchmark oriented analysis indicates potential improvement of adaptability and of sample efficiency: but these benefits have to be demonstrated using experimental raw data and real world testing.

7.2 Implications of the Research

The framework has implications for applications of reinforcement learning in which successful decisions are executable across multiple levels and transfer is necessary where uncertainty exists. Examples of possible domains are for robotics, autonomous systems and financial decision support. Under these conditions, ie., in uncertainty settings, the posterior uncertainty can be used to decide on the reuse, adapt or reject of source knowledge.

7.3 Future Work and Research Directions

Further investigations are needed for computational complexity, scalability and previous sensitivity. Reproducible experiments, with ablations, also, better posterior approximations, better mechanisms for adapting policies, and testing in more domains are important next steps. To ensure the practical viability and impact of the BHRL framework, the experiments need to be scaled up and carefully controlled real-world deployments are necessary.

REFERENCES

. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

. V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015.

. A. S. Vezhnevets et al., "FeUdal Networks for Hierarchical Reinforcement Learning," in Proc. 34th Int. Conf. Machine Learning, PMLR, vol. 70, pp. 3540-3549, 2017.

. Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in Proc. 33rd Int. Conf. Machine Learning, PMLR, vol. 48, pp. 1050-1059, 2016.

5. I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in Advances in Neural Information Processing Systems 29, pp. 4026-4034, 2016.
6. I. Osband, B. Van Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in Proc. 33rd Int. Conf. Machine Learning, PMLR, vol. 48, pp. 2377-2386, 2016.
7. Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," Nature, vol. 521, pp. 452-459, 2015.
8. D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484-489, 2016.
9. C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.
10. C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, nos. 3-4, pp. 279-292, 1992.
11. G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge Univ. Eng. Dept., Tech. Rep. CUED/F-INFENG/TR 166, 1994.
12. R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine Learning, vol. 8, pp. 229256, 1992.
13. R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," Artificial Intelligence, vol. 112, nos. 1-2, pp. 181-211, 1999.
4. A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," Discrete Event Dynamic Systems, vol. 13, pp. 4177, 2003.
5. T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in Advances in Neural Information Processing Systems 29, 2016.
6. T. Bayes and R. Price, "An essay towards solving a problem in the doctrine of chances," Philosophical Transactions of the Royal Society of London, vol. 53, pp. 370-418, 1763.
7. A. Gelman et al., Bayesian Data Analysis, 3rd ed. Boca Raton, FL, USA: CRC Press, 2013.
8. E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv:1012.2599, 2010.
9. M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," Journal of Machine Learning Research, vol. 10, pp. 1633-1685, 2009.
0. A. A. Rusu et al., "Progressive neural networks," arXiv:1606.04671, 2016.
1. S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, 2010.
2. R. Caruana, "Multitask learning," Machine Learning, vol. 28, pp. 41-75, 1997.
3. Y. Ganin et al., "Domain-adversarial training of neural networks," Journal of Machine Learning Research, vol. 17, no. 59, pp. 1-35, 2016.