Universiteit Leiden
The Netherlands

Data Science and Artificial Intelligence

Investigating the transferability of Reinforcement learning-based Hyperparameter optimisation methods

Jan Richtr

Supervisors:
Dr. J.N. van Rijn
Prof. Dr. Aske Plaat

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

dd/mm/yyyy

Abstract

This thesis replicates the Hyp-RL framework and evaluates its transferability to a new domain. It investigates whether the framework's ability to achieve strong results with fewer trials holds when applied to tune tabular reinforcement learning on varying environments. We compare two reinforcement learning-based hyperparameter optimisation methods, which select hyperparameter configurations from a discrete hyperparameter configuration space. One method uses the original DQN agent, and the other uses a newly adapted PPO agent for hyperparameter optimisation. These are tested against common baselines, i.e. Bayesian optimisation and random search. The experiments use trial budgets of 10 and 50. Results show that the DQN and PPO agents are better at achieving high returns with only a few trials, reaching 95% of the best return quickly across all environments. This is consistent with prior findings of the original paper. At low budgets, the DQN and PPO agents perform better in two of the evaluated environments, while in the others, the performance of all methods is similar. When the budget is increased, the relative ranking shifts and traditional baselines achieve comparable final returns, whereas in the environment with the highest reward stochasticity, performance remains unchanged. Overall, our findings suggest that the DQN and PPO agents are particularly effective under low trial budgets, while traditional methods perform better when more evaluations are possible.

Contents

1 Introduction

Hyperparameter optimisation (HPO) is the process of supporting the human in the loop in selecting the configuration that maximises a model's performance on a given task [HKV19]. In reinforcement learning, hyperparameters play a particularly critical role; the choice of values can make the difference between an agent that converges to a useful policy (the mapping from states to actions) and one that performs poorly or fails to improve [ELR23]. This sensitivity arises from the stochastic nature of reinforcement learning environments, the high variance of rewards, unstable learning dynamics, and the sequential nature of updates. All of which makes the performance of reinforcement learning agents heavily dependent on the chosen hyperparameters [ELR23].

Traditional HPO methods, such as random search [BB12] and Bayesian optimisation [SSW+16], have been widely applied to reinforcement learning [ELR23]. While these approaches can be effective, they often require relatively many trials compared to more informed and adaptive HPO methods [LJD+17]; each new task requires starting the search from scratch, evaluations can be noisy, and performance degrades sharply when the evaluation budget is small [LJD+17].

A possible alternative is reinforcement learning-based hyperparameter optimisation, which frames the HPO problem itself as a sequential decision-making task. Here, a reinforcement learning agent learns a policy for proposing hyperparameters, conditioned on observations of the current task and the outcomes of past configurations of both the current and training tasks. This approach offers several potential advantages: knowledge can be reused across tasks, experience can accumulate over many optimisation runs, and the search can adapt dynamically to partial results rather than following a fixed trajectory [JGS19]. Reinforcement learning-based hyperparameter optimisation has shown promise in supervised learning (tuning the hyperparameters of neural networks) [JGS19], but its effectiveness for optimising reinforcement learning agents themselves remains largely unexplored. This thesis will focus on a discrete configuration space consisting of 42 possible hyperparameter configurations, where each configuration corresponds to a specific choice of hyperparameter values. This thesis adapts the Hyp-RL framework [JGS19], which introduces a reinforcement learning-based hyperparameter optimisation method using a Deep Q-Network (DQN) agent and meta-features. Hyp-RL demonstrated quick convergence towards promising configurations when applied to tuning neural networks for supervised learning. An experiment consists of using one HPO method across 5 independent runs of a specified trial budget for a single task. This thesis does not propose a new framework; instead, it aims to reproduce and evaluate Hyp-RL when applied to a new domain, the tuning of tabular reinforcement learning agents [KLM96]. The methodology follows the same approach as Hyp-RL, but with an intentionally smaller experimental scope, focusing on a new domain.

In addition to reproducing the DQN agent from Hyp-RL, this thesis also has the novel contribution of adapting a Proximal Policy Optimisation (PPO) [SWD+17] agent to HPO. Unlike Hyp-RL, where the DQN agent tuned neural networks, both DQN and PPO agents here are applied to optimising reinforcement learning agents. The PPO agent uses the same setup as the DQN agent to ensure a fair comparison, allowing us to contrast the two approaches. Both the agents, will be selecting hyperparameter configurations from the discrete hyperparameter configuration space of size 42.

The goal of this thesis is to assess whether the advantages reported for Hyp-RL carry over when applied to reinforcement learning agents beyond the originally tested task. Specifically, we examine across three new environments whether the DQN agent retains its quick convergence

when tuning a SARSA tabular reinforcement learning agent (a classical on-policy method for learning action-value functions in tabular settings), whether a PPO variant can serve as an alternative reinforcement learning-based hyperparameter optimisation method, and how the relative performance of reinforcement learning-based hyperparameter optimisation methods and traditional baselines changes as the optimisation budget increases. Concretely, we compare methods under budgets of 10 and 50 trials, where a trial corresponds to one full training and evaluation of a SARSA agent with a particular hyperparameter configuration. In brief, DQN agents extend tabular Q-learning with deep neural networks to approximate value functions, and a PPO agent is a policy-gradient method designed for stable and efficient updates.

1.1 Research Questions

The primary goal of this study is to reproduce, extend, and evaluate the generalisability of the Hyp-RL framework (Jomaa et al., 2018) for tabular reinforcement learning agents. In this study, we investigate two reinforcement learning-based hyperparameter optimisation methods, which are two variants of the Hyp-RL framework, the original DQN agent and a novel PPO variant. Thus, the research question is:
How do the reinforcement learning-based hyperparameter optimisation methods perform when tuning SARSA agents under fixed evaluation budgets compared to standard baselines such as Bayesian optimisation and random search, in terms of final performance and convergence metrics? These metrics include (i) the number of trials required to reach 95% of the overall best return (trials to 95%), (ii) the fraction of runs that reach the 95% threshold within the first 40% trials (success rate at 40%), and (iii) the median number of configurations exceeding 95% of the best return within the first 40% trials (good-performing configurations at 40%).
The research question is addressed using 3 sub-questions:

1. To what extent can the original DQN agent from the Hyp-RL framework be applied to tuning SARSA agents, and how does its performance compare to Bayesian optimisation and random search baselines based on convergence metrics?

2. Under a fixed budget of 10 trials, how does the PPO variant of the Hyp-RL framework compare to the DQN agent in optimising SARSA agents across three unseen tabular environments, based on the final mean return?

3. When the budget increases from 10 to 50 trials, how do the relative performances and rankings of the DQN and PPO agents, Bayesian optimisation, and random search change, based on the final mean return and convergence metrics?

1.2 Thesis overview

The remainder of this thesis is organised as follows: Chapter 2 reviews related work on HPO in reinforcement learning; Chapter 3 dives deeper into the intuition and explanation of reinforcement learning-based hyperparameter optimisation methods and explains other important aspects required for the thesis. Chapter 4 describes the methodology, including the reinforcement learning-based hyperparameter optimisation methods; Chapter 5 describes the experimental design; Chapter 6 presents the results and their analysis; Chapter 7 discusses the findings in the context of existing

literature and interprets their implications; Chapter 8 concludes with a summary of the main contributions, limitations, and outlines potential avenues for future research.
The code and experimental data are available at https://github.com/Honzus/RL-HPO

2 Related Work

This section reviews key literature relevant to the thesis, focusing on reinforcement learning-based hyperparameter optimisation and its application to optimising tabular reinforcement learning agents. Despite an extensive literature search, reinforcement learning-based hyperparameter optimisation remains a relatively under-explored area, possibly due to the complexity of integrating reinforcement learning techniques into HPO frameworks or the novelty of the approach. Nevertheless, reinforcement learning has been commonly used for neural architecture search [BWL+24], most famously by Zoph et al. (2016) [ZL16], but also for example in NASNet [BGNR17] and MetaQNN [ZVSL18]. The following works provide the theoretical and empirical foundation for the proposed reinforcement learning-based hyperparameter optimisation framework and its evaluation, highlighting both the potential and challenges of this methodology.
Jomaa et al. (2019) [JGS19] propose the Hyp-RL framework, which models HPO as a Markov Decision Process. In their approach, a Deep Q-Network (DQN) agent dynamically selects hyperparameter configurations for supervised learning models, focusing on topological neural networks. The DQN agent learns a policy to navigate the hyperparameter space by leveraging meta-features, such as dataset size, feature count, and class imbalance, to generalise across datasets. Their experiments demonstrate that Hyp-RL outperforms traditional methods such as random search and Bayesian optimisation in terms of validation accuracy and the ability to achieve strong performance with few samples. Specifically, Hyp-RL achieves superior performance in benchmark datasets while requiring fewer evaluations, due to its ability to transfer learned policies across similar tasks. This transferability is particularly relevant to this thesis, as it aims to extend Hyp-RL by adapting its framework to optimise hyperparameters for tabular reinforcement learning agents and incorporating domain-specific meta-features to enhance performance in reinforcement learning environments. However, Jomaa et al.'s work is limited to supervised learning, leaving open the question of its applicability to the stochastic and dynamic nature of reinforcement learning tasks, which this thesis seeks to address.
Eimer et al. (2023) [ELR23] investigate the impact of HPO on reinforcement learning agents, comparing automated methods such as Bayesian optimisation and evolutionary algorithms against manual tuning. Their findings indicate that automated HPO consistently outperforms manual tuning, achieving higher normalised scores in standard reinforcement learning environments such as Brax [FFR+21] and ProcGen [CHHS20]. However, they highlight significant challenges in comparing HPO methods across reinforcement learning environments due to high variance in performance, driven by factors such as environment stochasticity and algorithm sensitivity. For instance, their experiments show that evolutionary algorithms are more robust to noisy environments but require more computational resources than Bayesian optimisation. Eimer et al. emphasise the need for robust and adaptive optimisation methods to improve reinforcement learning performance, supporting this thesis's hypothesis that reinforcement learning-based hyperparameter optimisation can address these challenges by dynamically exploring the hyperparameter space. Their work also underscores the importance of standardised evaluation protocols, which this thesis will incorporate

to ensure reliable comparisons between reinforcement learning-based hyperparameter optimisation methods and other HPO methods.

Parker-Holder et al. (2022) [PRS+22] describe automated reinforcement learning (AutoRL) as a bi-level optimisation problem: an outer loop chooses design decisions such as hyperparameters, architectures, or algorithms, while an inner loop trains the agent. They argue that reinforcement learning's brittleness and non-stationarity make automation, especially HPO, essential. The survey categorises HPO methods into static searches (grid and random search), multi-fidelity approaches (e.g. Hyperband, BOHB), model-based Bayesian optimisation using training curves, evolutionary or population-based schedules, and meta-gradient or black-box schemes that adapt hyperparameters during training. Key findings indicate that dynamic schedules outperform static ones due to changing data distributions, multi-fidelity evaluation reduces tuning cost, and robust evaluation across seeds using distribution-aware metrics (e.g. interquartile mean or optimality gap) is crucial given reinforcement learning's high variance. Open challenges include scaling to larger mixed search spaces, handling non-stationarity over time, transferring configurations across tasks, and creating standard benchmarks, some of which this thesis aims to address.

Henderson et al. (2018) [HIB+18] provide critical insights into the reproducibility challenges in deep reinforcement learning, emphasising the sensitivity of reinforcement learning algorithms to hyperparameters and environmental factors. Their study demonstrates that small changes in hyperparameters, such as learning rate or discount factor, can lead to performance differences (e.g., up to 15-34% in cumulative rewards) across environments such as MuJoCo [ZTL+25] tasks. They attribute this variability to factors including random seed differences, implementation details, and inconsistent evaluation protocols. For example, their experiments with Deep Deterministic Policy Gradient show that performance can vary significantly across runs with different seeds, even with identical hyperparameters. Henderson et al. advocate for standardised evaluation frameworks and robust HPO methods to mitigate these issues, aligning with the motivation for this thesis to develop a reinforcement learning-based hyperparameter optimisation approach that ensures consistent and reproducible optimisation of reinforcement learning agents. Their findings highlight the necessity of addressing hyperparameter sensitivity, which this thesis will tackle by leveraging reinforcement learning-based hyperparameter optimisation's ability to adaptively explore and exploit the hyperparameter space.

Other approaches have explored concepts related to this thesis, such as Hyperband [LJD+17], a bandit-based method that efficiently allocates resources to promising hyperparameter configurations through successive halving. While Hyperband does not directly employ reinforcement learning, its adaptive resource allocation strategy shares similarities with reinforcement learning's exploration-exploitation trade-off, offering insights into efficient hyperparameter search.

Additionally, Snoek et al. (2012) [SLA12] explore Bayesian optimisation for HPO in machine learning models, demonstrating its effectiveness in optimising complex hyperparameter spaces with fewer evaluations than grid or random search. Their work highlights the importance of modelling the hyperparameter response surface, which is relevant to reinforcement learning-based hyperparameter optimisation's use of learned policies to navigate similar spaces. While their focus is on supervised learning, their findings on the importance of uncertainty quantification in optimisation decisions provide insights for designing reinforcement learning-based hyperparameter optimisation frameworks that balance exploration and exploitation in reinforcement learning settings.

Recent studies have further explored meta-reinforcement learning for HPO, with a focus on leveraging environment meta-features to enhance optimisation efficiency. Wu et al. (2023) [WLC23]

introduce a meta-reinforcement learning framework that optimises hyperparameters by incorporating task-aware representations derived from environment characteristics, such as dataset properties and task-solving experience. Their approach demonstrates improved optimisation efficiency compared to baselines such as Bayesian optimisation, highlighting the value of meta-features in guiding hyperparameter selection. This work is conceptually relevant to this thesis, as it aligns with our use of meta-features to adapt reinforcement learning-based hyperparameter optimisation methods, where we extend the Hyp-RL framework to reinforcement learning contexts with the SARSA tabular reinforcement learning agent. While general HPO can be computationally intensive, Wu et al.'s method aims to mitigate this through meta-learning, which is similar to this thesis, which uses DQN and PPO agents with meta-learning.
Together, these works underscore the potential of reinforcement learning-based hyperparameter optimisation to address the limitations of traditional HPO methods in reinforcement learning contexts. They also highlight challenges such as performance variance, computational efficiency, and the need for standardised evaluation.

2.1 Critique of Hyp-RL

The original Hyp-RL framework employs a DQN agent, which is inherently designed to manage high-dimensional data. However, the explicit motivation for selecting this precise algorithm is unclear.
When evaluating HPO in general, the number of hyperparameters subject to optimisation is often limited. From a reinforcement learning perspective, these problems can realistically be categorised as low-dimensional (tens of hyperparameters), thus enabling the application of standard tabular reinforcement learning methods, such as Q-learning [SB98]. An advantage of the tabular methods lies in their interpretability, as neural networks often function as 'black boxes'.
It is important to mention that Hyp-RL incorporates meta-features into its state representation, which expands the dimensionality of the state space. Furthermore, the formulation of the action space being all 'explored' hyperparameter configurations can also lead to a combinatorial explosion, potentially posing challenges for tabular methods. Nevertheless, this could be an interesting, simpler avenue to think about.

3 Background

3.1 Hyperparameter Optimisation for Supervised Learning

In this section, we focus on HPO in the context of vanilla supervised machine learning, where the objective is to learn from labelled data by minimising a loss function.
Hyperparameters, unlike model parameters, are set before training and greatly impact the performance, robustness and generalisability of machine learning models [HKV19]. On the other hand, model parameters are learned during the training phase. As a result, the field of HPO has risen within automated machine learning (AutoML), and develops algorithms/methods that optimise hyperparameters of machine learning models [HKV19]. Intuitively, HPO is the systematic selection of hyperparameter configurations to maximise a model's performance, or equivalently, minimise a given loss function. Some commonly used HPO techniques/methods include grid search, Bayesian optimisation or random search [HKV19, BB12, SSW+16].

Formally, a machine learning model learns a mapping from the input space X (the space of all possible inputs) to the output space Y (the space of all possible output) [HKV19]:

$$f(\mathbf{x}, \theta, \lambda) : X \to Y \tag{1}$$

where θ are the learned model parameters, $\lambda \in \Lambda$ is a fixed hyperparameter configuration from the set of all hyperparameters, and $\mathbf{x}$ is an input vector in the input space X. In the following equations, y denotes the corresponding label of the input vector $\mathbf{x}$ in the output space Y. The model learns by minimising some arbitrary loss function $\mathcal{L}$ on the training set. Thus, training equates to finding the optimal model parameters θ^* for a given hyperparameter configuration, by minimising the given loss function over N training samples [JGS19, BWL+24, HKV19] :

$$\theta^*(\lambda) = \arg\min_{\theta} \left(\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}, \theta, \lambda), y) \right) \tag{2}$$

Training can be described as a function that finds the optimal model parameters for a given hyperparameter configuration in the training set. HPO can then be thought of as minimising the same loss function for the optimal model parameters for all hyperparameter configurations. To find this optimal hyperparameter configuration λ^*, hyperparameter selection is the outer loop, and the training phase is the inner loop. The hyperparameter configuration is then evaluated on the validation set after the model parameters are found [JGS19, BWL+24, HKV19]:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \left(\frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(f(\mathbf{x}, \theta^*(\lambda), \lambda), y) \right) \tag{3}$$

where M is the number of validation samples in the validation set [HKV19]. Equations 2 and 3 describe the standard HPO process for supervised machine learning tasks, where the objective is to minimise a given loss function. In contrast, for reinforcement learning, the objective is to maximise the cumulative reward $r(\theta^*(\lambda), \lambda)$ over a predefined number of episodes, since we are not dealing with labelled datasets as in supervised machine learning. The cumulative reward reflects the SARSA agent's long-term performance [ELR23]. Moreover, in reinforcement learning, hyperparameters are tuned based on the cumulative reward of evaluation episodes rather than in a separate validation phase, as held-out validation samples are not used. This requires exploring the high-dimensional hyperparameter space Λ, which poses computational challenges due to its size and complexity [ELR23].

3.2 Reinforcement Learning

Reinforcement learning is a field within machine learning which focuses on training agents to make sequential decisions by interacting with an environment [KLM96]. Through interacting with the environment, the agent's objective is to maximise cumulative reward, which is contrary to conventional supervised learning that uses labelled data [KLM96]. Reinforcement learning problems are generally formulated using Markov Decision Processes, which provide a mathematical framework for modelling sequential decision-making [KLM96].

A Markov Decision Process is defined as a tuple (S, A, P, R, γ). Here S is the set of all possible states, A is the set of all possible actions the agent can take, $P(s'|s, a)$ is the state transition

probability distribution describing the likelihood of moving to state s' from state s after taking action a. More generally, one can think of the distribution as describing transitions between any two states. The state transition probability distribution is also referred to as the transition function τ. $R(s, a, s')$ is the reward function returning the expected reward for a given transition, and $\gamma \in [0, 1]$ is the discount factor balancing immediate and future rewards [KLM96, SB98].
At any discrete time step t, the agent observes the state $s_t \in S$, selects an action $a_t \in A$, and receives a reward $r = R(s_t, a_t, s_{t+1})$. The environment transitions to a state $s_{t+1} \sim P(\cdot|s_t, a_t)$. The process repeats until termination; the process from start to end is called an episode. The agent aims to maximise the expected cumulative discounted reward [SB98, KLM96]:

$$G = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{4}$$

The agent seeks an optimal policy $\pi^*(a|s)$ that maximises G. To do so, it balances exploration and exploitation.
Reinforcement learning agents learn optimal policies via value-based, policy-based, or actor-critic methods. Value-based methods estimate the value of state-action pairs through the Q-function [SB98]:

$$Q(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right] \tag{5}$$

For small or discrete state spaces, tabular algorithms such as Q-Learning or SARSA are applicable [KLM96]. For large or continuous spaces, deep reinforcement learning algorithms such as DQN approximate $Q(s, a)$ with neural networks [SB98].
Conversely, policy-based approaches directly optimise the policy $\pi(a|s)$ using gradient ascent on the expected cumulative reward [SB98]:

$$J(\theta) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{6}$$

Tabular policy-gradient methods (e.g. REINFORCE) or deep reinforcement learning variants (e.g. PPO) can be used depending on the problem scale [SWD+17, SB98].
Actor-critic methods combine both paradigms: an actor learns a parametrised policy $\pi_\theta(a|s)$ while a critic estimates a value function to reduce variance in policy updates, yielding improved stability. [SB98]

3.3 Motivation for reinforcement learning-based hyperparameter optimisation

The motivation for using reinforcement learning-based hyperparameter optimisation lies in its ability to model HPO as a sequential decision-making problem, where states (evaluations of hyperparameter configurations) are inherently linked to prior states due to the iterative nature of the HPO process [HKV19]. Unlike systematic methods such as grid search, which evaluate models over a predefined set of hyperparameter combinations, reinforcement learning exploits this sequential structure to optimise hyperparameters systematically [JGS19]. Compared to other non-systematic methods

like Bayesian optimisation, which treat hyperparameter evaluations independently and fail to leverage relationships between configurations [SSW+16], this sequential approach is particularly advantageous for reinforcement learning agents whose stochastic behaviour amplifies the need for adaptive optimisation. This should enable reinforcement learning-based hyperparameter optimisation methods to efficiently navigate complex hyperparameter spaces, improving performance and speed of convergence to promising configurations [JGS19].

4 Methodology

4.1 Reinforcement learning-based hyperparameter optimisation environment formulation

This thesis follows the same environment formulation and Markov Decision Processes as in Hyp-RL [JGS19]. States $s \in S$ are defined as tuples combining dataset meta-features D and the history of evaluated hyperparameter configurations and their rewards [JGS19]:

$$S = D \times (\Lambda \times R)^* \tag{7}$$

The action space A is the hyperparameter grid Λ, that is $A = \Lambda$, allowing the agent to select discrete hyperparameter configurations. Then the reward function R is defined as the cumulative reward of the agent in an environment configured with an action a on a dataset D. By which, the agent tries to pick actions that maximise rewards [JGS19].

$$R(D, a) = r(D, a) \tag{8}$$

Moreover, the transition function τ (see 3.2), as defined in Hyp-RL, appends the latest hyperparameter configuration and its performance (cumulative reward of the SARSA agent) to the state history, updating the state to reflect the new evaluation. The episode terminates when the evaluation budget (the total number of actions in the action space) is exceeded, or the optimising agent (PPO or DQN in the context of this thesis) selects the same action twice, ensuring efficient resource use.

4.2 Reinforcement learning Algorithms Used

4.2.1 Deep Q-Network

The Deep Q-Network (DQN) algorithm extends traditional Q-Learning for large or continuous state spaces [SB98]. It approximates the value function, $Q(s, a)$, with a neural network with parameters θ, written as [KLM96, SB98]:

$$Q_\theta(s, a) \tag{9}$$

This function estimates the expected cumulative reward of choosing an action a in a state s. The DQN agent is trained by minimising the Temporal-Difference (TD) error, which measures the difference between the current estimate of the Q-value and the target Q-value derived from the Bellman optimality equation:

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim D_{replay}} \left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a) \right)^2 \right] \tag{10}$$

Here, the environment formulation is the same as described in Section 4.1, r is the reward (performance achieved by evaluating a given hyperparameter configuration) taking action a (a discrete hyperparameter configuration) in state s (meta-features and history of previously evaluated hyperparameters of the current dataset), γ is the discount factor, and θ^- are the parameters of a target network, which is updated periodically to stabilise learning.
To reduce correlations between sequential experiences and improve sample efficiency, an experience replay buffer D_{replay} is used. It stores past transitions (s, a, r, s') and samples them uniformly during training. This allows the DQN agent to learn from diverse past experiences and prevents instability caused by highly correlated updates.

4.2.2 Proximal Policy Optimisation

Proximal Policy Optimisation (PPO) is an actor-critic policy-gradient algorithm which makes stable and efficient policy updates [SWD+17]. The environment formulation is the same as was described in Section 4.1.
PPO consists of two components.

1. Actor - a stochastic policy $\pi_\theta(a|s)$, a neural network parametrised by θ, which selects actions.
2. Critic - a value function $V_w(s)$, a neural network parametrised by w, estimating the expected cumulative reward of being in state s.

The actor is trained to maximise a clipped surrogate objective, which is an objective function that limits how much the policy can change in a single update to prevent instability [SWD+17]:

$$L^{\text{clip}}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \right], \tag{11}$$

here, A_t is the advantage estimate representing how much better the selected action is compared to the expected value of the current state. The policy ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ measures how much the probability of taking action a_t in the state s_t has changed under the new policy compared to the old policy. θ_{old} are the policy parameters before the update, and ϵ is a small positive constant that limits the maximum allowed change in the probability a policy has in selecting a specific action. The clip() function constrains the policy ratio to stay within $(1-\epsilon, 1+\epsilon)$, preventing large updates that could destabilise training.
The combined loss updates both the actor and the critic simultaneously [SWD+17]:

$$L(\theta, w) = L^{\text{clip}}(\theta) - c_1 \mathbb{E}_t \left[(V_w(s_t) - \hat{V}_t)^2 \right] + c_2 H(\pi_\theta(\cdot|s_t)), \tag{12}$$

where, c_1 and c_2 weigh the critic's value loss (error between predicted and computed state values $V(s)$) and the entropy bonus (extra reward encouraging the agent to explore more diverse actions) [SWD+17]. $\hat{V}_t$ is the target value used to train the critic, and $H(\pi_\theta)$ denotes the policy's entropy, a measure of randomness in the agent's action selection, higher entropy encourages more exploration [SWD+17].
Clipping the policy update in PPO prevents large changes that could destabilise learning [SWD+17]. In the context of HPO, this allows the agent to safely explore and evaluate different hyperparameter configurations while steadily improving its policy for selecting high-performing hyperparameter configurations.

4.3 Agent Architectures and Training

Within this thesis, we adapt and implement two reinforcement learning-based hyperparameter optimisation methods: the original DQN agent and a novel PPO agent.

DQN Agent

The DQN agent follows the same architecture as in the original Hyp-RL framework [JGS19]. It employs a neural network combining Long Short-Term Memory (LSTM) layers with multilayer perceptrons and ReLU activations to approximate Q-values, which estimate the expected cumulative reward of selecting a given hyperparameter configuration based on the current state (dataset meta-features and previously evaluated hyperparameters). The LSTM processes the sequence of past hyperparameter configurations and rewards, capturing temporal dependencies in the search process. Its initial hidden state is initialised using dataset meta-features through a learnable linear transformation, allowing the policy to adapt to different datasets from the start. The LSTM output is passed through a fully connected layer to produce Q-values for each hyperparameter configuration in the discrete grid, enabling action selection through an ϵ-greedy policy.
To ensure stable learning, the agent uses an experience replay buffer that stores past transitions (s, a, r, s') and samples mini-batches during training to reduce correlations between updates. The model is trained using the Adam optimiser to minimise the TD error. The agent balances exploration and exploitation using an ϵ-greedy strategy: with probability ϵ, it selects a random action to explore, otherwise it chooses the action with the highest predicted Q-value. The value of ϵ decays over time, gradually making the agent more exploitative. Dataset cycling is used to simulate varied tasks, helping the agent generalise and potentially transfer learned hyperparameter strategies across different reinforcement learning environments.

PPO Agent

The PPO agent uses the same neural architecture as the DQN agent for both the actor and the critic. Unlike the DQN, it employs a PPO buffer to store full trajectories and compute Generalised Advantage Estimates (a method to reduce variance in policy gradient updates by smoothing advantage estimates across time steps), which are normalised to stabilise training and mitigate issues with missing or unstable values (NaNs) [SWD+17]. Such NaNs can appear when value estimates diverge early in training or when the variance of the advantages becomes very small, causing numerical instability. Policy updates are carried out over multiple epochs using mini-batches, and the clipped surrogate objective is optimised with the Adam algorithm to ensure stable yet effective policy improvements. Actions are sampled from a softmax-scaled categorical distribution, introducing controlled stochasticity that supports exploration of alternative hyperparameter configurations. Dataset cycling follows the same procedure as for the DQN agent.

4.4 Meta-features

Meta-features are critical components of state representation for reinforcement learning-based hyperparameter optimisation. They form part of the environment formulation and define how states are represented. Thanks to meta-features, agents can make more informed decisions within the optimisation environment [JGS19]. Meta-features are descriptive characteristics of the reinforcement

learning environment, such as the size of the state space, the number of possible actions, or the structure of the reward functions [HAMS22].
In traditional machine learning datasets, meta-features typically describe dataset properties, such as the number of features or class imbalance. However, reinforcement learning environments are dynamic and partially observable, with data characteristics that are largely unknown without extensive exploration [SB98]. This makes traditional meta-features less effective. Instead, this thesis uses environment-based meta-features that capture structural or statistical properties of the reinforcement learning task, such as the state space size or the cardinality of the action space [BVL+25]. These meta-features enable the reinforcement learning-based hyperparameter optimisation methods to generalise HPO strategies across different environments or datasets, enhancing transferability.
Since many environment-based meta-features are categorical, they are encoded using one-hot encoding to create numerical representations suitable for neural network inputs. Numerical meta-features, such as action space size, are included directly or normalised. The encoded meta-features are concatenated with the hyperparameter configuration history and cumulative rewards to form the state vector, which is processed by the DQN and PPO agents' neural networks [JGS19].
The table of meta-features used in this thesis can be seen below.

Meta-feature	Description
Observation Space Type	Type of observation space, encoded as an integer using OBS_SPACE_TYPES mapping
Observation Space Shape	Unique SHA-1 hash of the shape of the observation space
Observation Space Data Type	Data type of observation space, encoded as an integer using DTYPE_MAP
Is image observation?	1 if observation space is a Box with 2+ dimensions, 0 otherwise
State Space Size	Number of states if observation space is Discrete, 0 otherwise
Observation Space Bound Low	Lower bound of observation space (Box only), multiplied by 1000, -999999 for -inf, or None if unavailable
Observation Space Bound High	Upper bound of observation space (Box only), multiplied by 1000, 999999 for inf, or None if unavailable
Action Space Type	Type of action space encoded as an integer using ACTION_SPACE_TYPES mapping
Is Action Space Discrete?	1 if action space is Discrete, 0 otherwise
Action Space Shape	Number of actions if Discrete, otherwise hashed shape of action space if Box
Action Space Bound Low	Lower bound of action space (Box only), multiplied by 1000, -999999 for -inf, or None if unavailable
Action Space Bound High	Upper bound of action space (Box only), multiplied by 1000, 999999 for inf, or None if unavailable
Reward Range Minimum	Minimum reward value, multiplied by 1000, -999999 for negative, or 0 if unknown
Reward Range Maximum	Maximum reward value, multiplied by 1000, 999999 for infinity, or 0 if unknown
Reward Density	Reward density sparse, dense, unknown, encoded as 0, 1, or 2 using REWARD_DENSITY mapping
Is episodic?	1 if environment has a defined maximum episode length, 0 otherwise
Max Episode Steps	Maximum number of steps per episode, 0 if not episodic
Is goal-based?	1 if environment name suggests a goal-based task, 0 otherwise
Available Render Modes	Bitmask of supported render modes using RENDER_MODES mapping
Is deterministic?	1 if deterministic, 0 if non-deterministic, 2 if unknown
Is time-dependent?	1 if environment name suggests time-dependent dynamics, 0 otherwise

Table 1: Meta-features used for state representation in reinforcement learning-based hyperparameter optimisation and their encodings.

5 Experimental Setup

This thesis investigates the transferability of reinforcement learning-based hyperparameter optimisation methods to tabular reinforcement learning tasks, comparing their performance against traditional HPO baselines. The methodology evaluates both the efficiency and effectiveness of reinforcement learning-based hyperparameter optimisation methods, focusing on their ability to locate high-return hyperparameter configurations across diverse tabular reinforcement learning environments. Results are collected for individual agents.

1. **Environment selection:** Tabular reinforcement learning environments are selected from Gymnasium's maintained suite (e.g. CliffWalking-v0, Taxi-v3, FrozenLake-v1), compatible open-source counterparts (e.g. Sepsis/ICU-Sepsis-v2), and custom environments (e.g.

ComplexMaze-v0). All environments use the Gymnasium library [TKT+24] to ensure consistent interfacing.

2. **SARSA agent implementation:** The SARSA tabular reinforcement learning agent is implemented with three tunable hyperparameters: learning rate (α), discount factor (γ), and exploration rate (ϵ). Policies follow an ϵ-greedy strategy, and value functions are stored in exact tables

3. **Baseline HPO methods:** Bayesian optimisation and random search are implemented using the Optuna library, operating on the same three-dimensional hyperparameter search space of the SARSA agent.

4. **Reinforcement learning-based hyperparameter optimisation methods:** The original DQN agent from Hyp-RL is adapted to optimise tabular reinforcement learning. A PPO agent is developed with the same environment formulation as the DQN agent.

5. **Offline data generation:** For each environment, the SARSA agent is run on 42 hyperparameter configurations (Cartesian product of $\alpha \in \{0.01, 0.02, 0.05, 0.1, 0.3, 0.5\}$ and $\epsilon \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, with $\gamma = 0.99$ fixed). Each configuration is executed with five random seeds for 10,001 episodes (20,001 for FrozenLake-8x8). After training, ϵ is set to 0 and the agent is evaluated for 200 episodes with fresh seeds. Environment meta-features and performance data are collected to form the offline training dataset for reinforcement learning-based hyperparameter optimisation methods.

6. **Training reinforcement learning-based hyperparameter optimisation methods:** DQN and PPO agents are trained on the offline datasets collected for each environment.

7. **Evaluation in test environments:** The trained reinforcement learning-based hyperparameter optimisation methods are used to optimise the SARSA agent in three previously unseen environments (ComplexMaze-v0, SimpleMaze-v0, Sepsis/ICU-Sepsis-v2). Each method is evaluated with 10 optimisation trials, repeated across five independent runs. Experiments are then repeated with 50 trials instead.

8. **Baseline evaluation in test environments:** Bayesian optimisation and random search are used to optimise the SARSA agent in the same three test environments under identical conditions (10 and 50 trials, five seeds per environment).

9. **Performance metrics:** Evaluate every HPO method using:

   (a) Final mean episodic return of the best configuration in each run,

   (b) Interquartile range (IQR) of the best return,

   (c) Standard deviation (SD) of the best return achieved,

   (d) Number of trials required to reach 95% of the best overall return,

   (e) Success rate (fraction of runs reaching the 95% threshold within the first 40% of trials),

   (f) Median number of good-performing configurations (exceeding 95% of the best return within the first 40% of trials).

These metrics jointly capture final performance, stability, and convergence speed, directly addressing the research objectives.

5.1 Test environments

The performance of the four HPO methods is evaluated on three unseen environments: SimpleMaze-v0, ComplexMaze-v0, and Sepsis/ICU-Sepsis-v2. The first two are custom grid-world environments inspired by open-source implementations such as SimpleGrid.
SimpleMaze-v0 is a deterministic 4×4 grid-world with 16 states. The agent can take four actions: up, down, left, or right. Rewards are sparse: the agent receives $+1.0$ for reaching the terminal state and 0 otherwise, giving a maximum cumulative reward of 1.0.
ComplexMaze-v0 is a stochastic 5×5 grid-world with the same action set as SimpleMaze-v0. In addition to the terminal reward of $+1.0$, two special states provide a bonus reward of $+0.5$ with probability 0.2, yielding a maximum cumulative reward of 2.0.
Sepsis/ICU-Sepsis-v2 is an open-source reinforcement learning environment simulating sepsis treatment in an intensive care unit [CGT24]. The agent selects between approximately 5 and 25 discrete actions, depending on 750-1500 patient states, to maximise patient recovery. Rewards are sparse: $+1.0$ for recovery, -1.0 for death, and 0 otherwise. High stochasticity due to patient variability makes this the most challenging environment for tabular reinforcement learning. There is a default maximum of 20 steps per episode.
All experiments were run with five independent seeds, consisting of 10 and 50 trial budgets per seed. For the DQN agent, a linear ϵ-decay schedule from 0.7 to 0.1 was used. The PPO agent applied a linear entropy-coefficient decay from 0.7 to 0.1. These schedules were reset at the start of each run.

6 Results

Figure 1 presents a heatmap of returns for a SARSA agent in the Cliffwalking-v0 environment, parametrised by exploration rate (ϵ) and learning rate (α). The visualisation shows substantial variation in performance across the hyperparameter space, with maximum returns consistently occurring in the low-ϵ/low-α region. This pattern indicates a strong interaction effect between

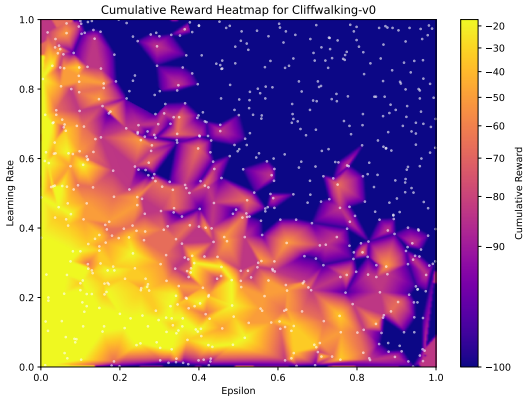
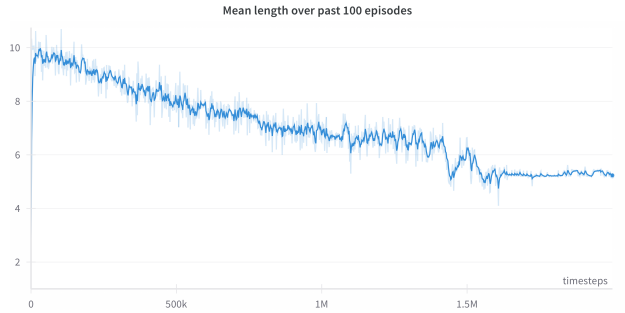


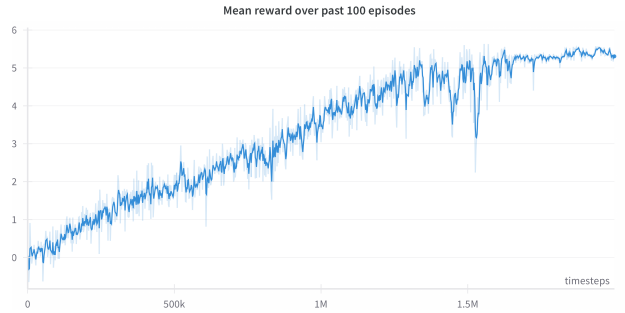
Figure 1: Heatmap showing how the cumulative reward of a SARSA agent varies with learning rate and exploration parameter (epsilon), with discount factor fixed to 0.99

exploration and learning rate: conservative learning strategies coupled with limited exploration yield optimal performance in this environment.
These observations motivate the application of the Hyp-RL framework, which reformulates HPO as a sequential decision process. The method identifies high-reward trajectories through the hyperparameter space (conceptually represented as paths across the heatmap) that maximise the sum of returns. When applied to new environments, the reinforcement learning-based hyperparameter optimisation methods first explore, then gradually converge towards the trained trajectories. This balanced approach leverages prior policy knowledge while acquiring environment-specific response surface characteristics to prevent premature convergence to suboptimal regions.
While the heatmap highlights a clear optimal region, this finding is specific to Cliffwalking-v0. In smoother or noisier environments, the same low-ϵ/low-α regime may not generalise. Additionally, because the analysis uses a discretised hyperparameter grid, local optima between grid points may be missed. Nonetheless, the heatmap provides a useful diagnostic, illustrating why sequential decision-making approaches to HPO can capture meaningful structure in the search space, while also highlighting the limitations of grid-based approximations.

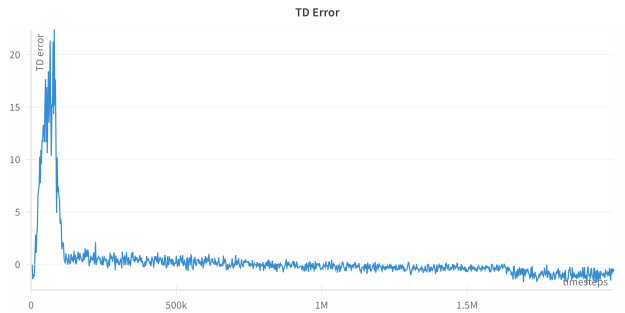
6.1 RQ1: Reproduction



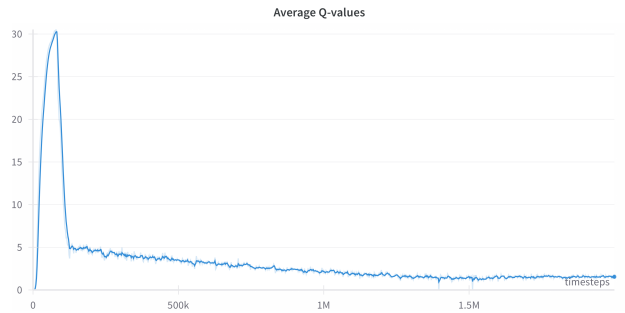
(a) Mean episode length of the DQN agent, calculated over a rolling window of 100 episodes and smoothed with a factor of 0.5.



(b) Mean reward of the DQN agent, calculated over a rolling window of 100 episodes and smoothed with a factor of 0.5.



(c) TD error of the DQN agent at each time step.



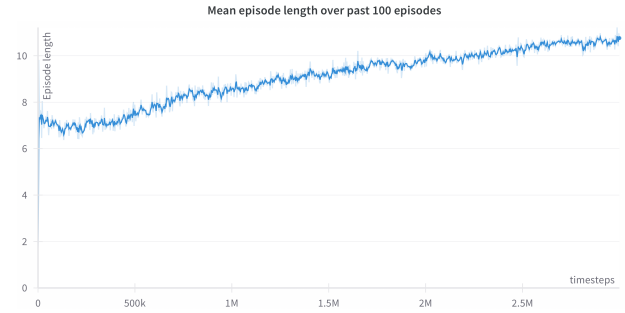
(d) Mean of all Q-values estimated by the DQN agent at each training time step.

Figure 2: Training behaviour of the DQN agent adapted from the Hyp-RL framework for the hyperparameter optimisation of a SARSA agent. Across all four metrics: mean reward, mean episode length, TD error, and Q-values, the agent shows consistent improvement and convergence during training, indicating successful learning.

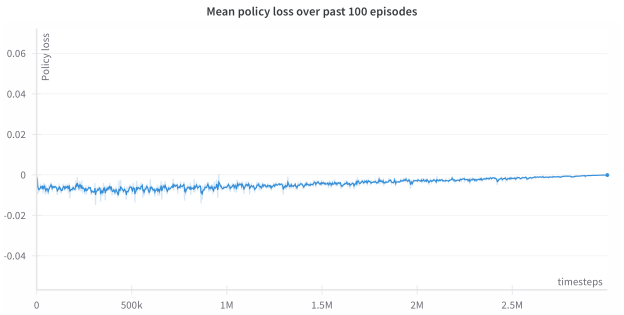
Figure 2 shows the behaviour of the adapted DQN agent during training over 2 million time-steps. Panel a) shows the rolling mean episodic reward increasing steadily from an initial value

of approximately 0 to 5.3, with a modest slope reflecting consistent progress in selecting effective hyperparameter configurations, despite variance from adapting to diverse dataset-specific optima. Panel b) displays the average episode length decreasing from 10 to 5 steps, indicating more efficient decision-making as the agent converges to a subset of optimal hyperparameters from 42 possible actions, with minor fluctuations due to occasional exploration of new configurations. Panel c) depicts the TD error, which peaks at 20 early in training due to random exploration, then declines to near 0 by 500,000 time-steps, stabilising with minor fluctuations, reflecting convergence to accurate Q-value predictions. Panel d), showing the Q-value curve mirrors this trend, with an initial overestimation spike (Q-values are inflated from exploration) followed by a decline and oscillations toward stability, driven by epsilon decay and increased experience. This initial overestimation is a common occurrence among DQN agents and is the result of taking the max operator of early noisy observations [SB98, KLM96]. The use of a Cartesian grid for hyperparameter configurations enables the agent to isolate the causal effect of single hyperparameter changes, yielding interpretable Q-table gradients and robust policy updates despite stochasticity. Together, these graphs show that the Hyp-RL framework effectively learns an optimisation policy after the domain shift, addressing the first part of the first research sub-question.

(a) Mean reward of the PPO agent, calculated over a rolling window of 100 episodes and smoothed with a factor of 0.5.



(b) Mean episode length of the PPO agent, calculated over a rolling window of 100 episodes and smoothed with a factor of 0.5.



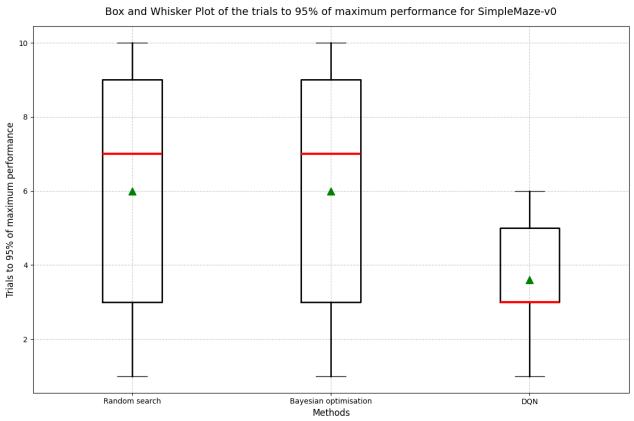
(c) Policy loss of the PPO agent, calculated over a rolling window of the previous 100 episodes and smoothed with a factor of 0.5.

Figure 3: Training curves of the PPO agent adapted for hyperparameter optimisation of a SARSA agent. (a) Mean reward curve. (b) Mean episode length curve. (c) Policy loss (clipped surrogate objective) curve, showing convergence. These curves confirm that the PPO agent successfully learns.

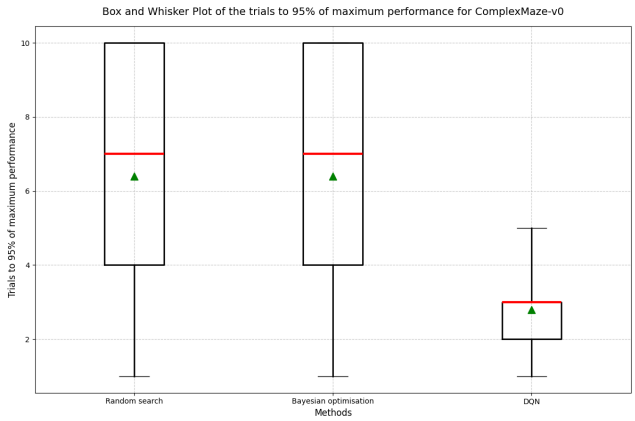
Figure 3 shows the training behaviour of the PPO agent over 3 million time-steps. Panel a) shows the rolling mean episodic reward first jumps to approximately 4 and then rises steadily from approximately 4 to 12, reflecting successful learning of effective hyperparameter configurations,

with minor fluctuations due to adaptation to diverse training environments. Panel b) displays the average episode length increasing from approximately 7 to 11 steps, indicating sustained interactions as the agent explores more promising hyperparameter configurations, in contrast to the DQN agent's focus on fewer optimal configurations. Panel c) depicts the policy loss, which exhibits high initial variability, peaking at -0.01, then stabilising near 0 by 2.5 million time-steps with tiny fluctuations, reflecting convergence to a stable action distribution within PPO's clipped surrogate objective. Additionally, the value loss (Appendix) initially rises to approximately 2.5 due to random exploration, then gradually declines without fully flattening, suggesting continuous improvement in the value network's predictions. The approximate KL divergence (Appendix) shows minor fluctuations near zero with occasional sharp drops, confirming stable policy updates close to prior policies. The entropy decreases from approximately 3.0 to 1.0, indicating a shift from exploration to a more deterministic policy. These trends collectively demonstrate that the PPO variant effectively learns an HPO policy for the tuning of a SARSA agent [SWD+17, KLM96], establishing it as a valid counterpart for comparative analyses in later research questions. Compared to the DQN agent, the PPO agent adapts more cautiously to diverse environments, indicated by a higher episode length, which may come at the cost of slower convergence.

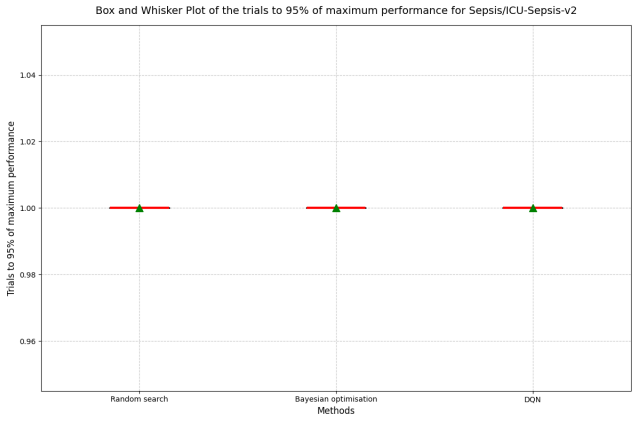
To assess convergence speed, we evaluated the number of trials required for each method to achieve 95% of its best-found return within a 10-trial budget, as shown in Figure 4. In SimpleMaze-v0 (panel a), the DQN agent reaches the 95% threshold in a median of 3.0 trials (IQR = 2.0), outperforming random search (median = 7.0, IQR = 6.0) and Bayesian optimisation (median = 7.0, IQR = 6.0), demonstrating superior ability to locate promising configurations quickly. In ComplexMaze-v0 (panel b), random search and Bayesian optimisation achieve the threshold the same again (median = 7.0, IQR = 6.0, range = [1,10]), and are outperformed by the DQN agent (median = 3.0, IQR = 1.0, range = [1,5]). DQN's tighter IQR, lower median and narrower range indicate good consistency, making it more reliable for locating promising configurations in familiar complex environments. In Sepsis/ICU-Sepsis-v2 (panel c), all methods reach the threshold in a median of 1.0 trial (IQR = 0.0) due to the flat reward surface. Overall, the DQN agent consistently achieves the lowest median and mean, along with low variance, which comes from the learned Q-values generalising to unseen environments. This reinforces its advantage in using few samples to find good configurations, as originally reported by [JGS19], and addresses the first research sub-question.



(a) SimpleMaze-v0: DQN attains the smallest median number of trials to reach 95%.



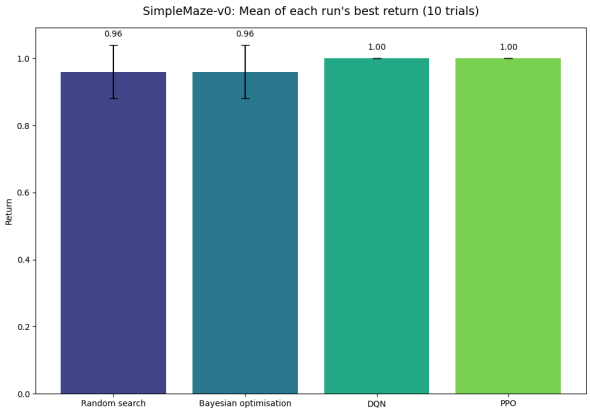
(b) ComplexMaze-v0: DQN attains the smallest variance of trials to reach 95%.



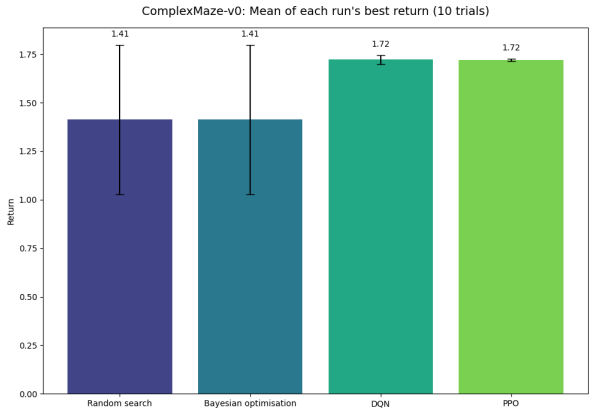
(c) Sepsis/ICU-Sepsis-v2: flat reward surface yields identical median of 1 trial for all methods.

Figure 4: Trials required to reach 95% of the best return under the 10-trial budget. Boxes show median and IQR across five independent seeds; whiskers extend to min–max. Lower values indicate a more efficient search of promising configurations.

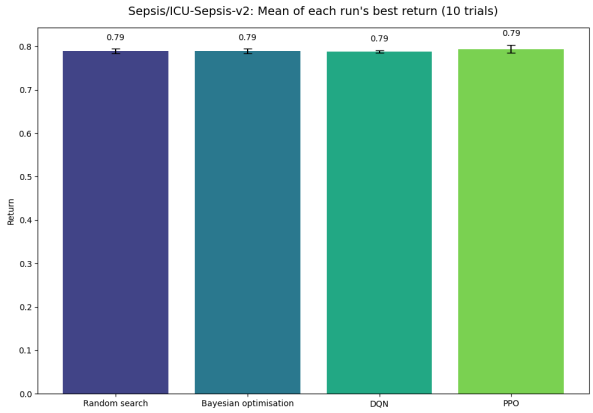
6.2 RQ2: 10-Trial Performance Comparison



(a) SimpleMaze-v0



(b) ComplexMaze-v0



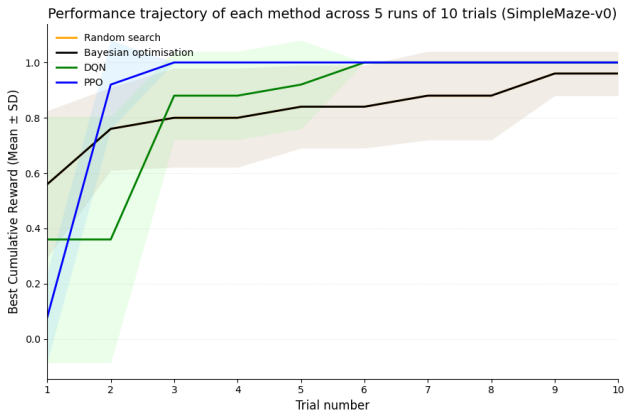
(c) Sepsis/ICU-Sepsis-v2

Figure 5: Final mean return (mean ± SD, 5 seeds) after exactly 10 trials for the three unseen environments.

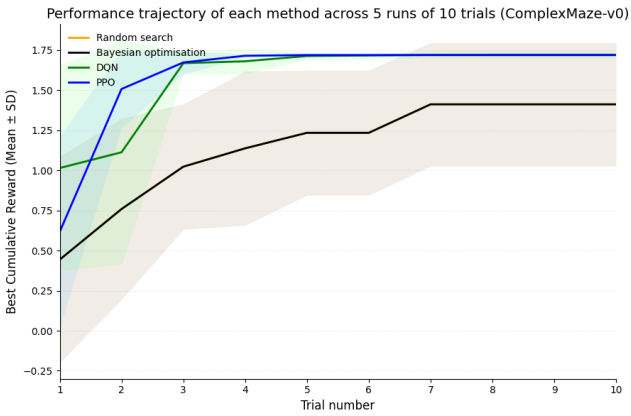
To address the second research sub-question, we evaluated the final mean return of each HPO method after 10 trials, averaged over 5 independent runs, across three unseen environments, as shown in Figure 5. In SimpleMaze-v0 (panel a), only the DQN and PPO converge to the maximum return of 1.00 ± 0.00, while both random search and Bayesian optimisation achieve a lower and less stable return of 0.96 ± 0.08. In ComplexMaze-v0 (panel b), PPO performs best with a mean return of 1.72 ± 0.01, followed closely by DQN (1.72 ± 0.01). The difference between the two methods is near indistinguishable, and their small deviations indicate good consistency. Random search and Bayesian optimisation achieve the same reward again (1.41 ± 0.33), trailing with higher deviations and lower returns. The baselines underperform due to the inability to explore the hyperparameter search space sufficiently. This is why Bayesian optimisation behaves exactly like random search, as the Gaussian process does not find stable patterns to exploit and is forced to keep exploring [SSW+16]. In Sepsis/ICU-Sepsis-v2 (panel c), all methods yield nearly identical returns of 0.79 ± 0.005 reflecting the flat reward surface.

The DQN and PPO agents perform well in ComplexMaze-v0 and SimpleMaze-v0, answering RQ2 by consistently finding high-return hyperparameters in structured environments. The demonstrated advantage of the reinforcement-learning-based methods over baselines suggests that for small

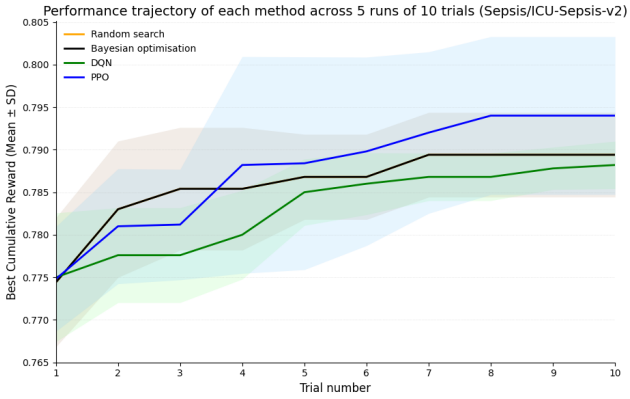
budgets, the baseline methods require more trials to achieve comparable final mean returns. Generally, the small three-environment suite may limit conclusions about performance in more varied reward settings.



(a) SimpleMaze-v0



(b) ComplexMaze-v0



(c) Sepsis/ICU-Sepsis-v2

Figure 6: Best-so-far mean episodic return across the 10-trial budget (mean ± SD, 5 seeds). Panel (a) SimpleMaze-v0, (b) ComplexMaze-v0, (c) Sepsis/ICU-Sepsis-v2.

To evaluate how quickly each method approaches peak performance within the 10-trial budget, we tracked the best-so-far mean episodic return across all runs at each trial (Figure 6).
For SimpleMaze-v0, PPO showed the greatest improvement magnitude, starting from the lowest initial returns (likely due to noisy evaluations). Bayesian optimisation exhibited equal performance to random search, indicating slow, consistent progress. However, neither consistently achieved a maximum return of 1.0 across all runs, leading to slightly lower convergence. Random search and Bayesian optimisation both displayed significant standard deviation in their whole trajectory, reflecting their inherent instability. The most stable methods was the PPO agent, which is the result of its stable update mechanism [SWD+17]. The DQN agent began with slightly higher initial returns, but also showed consistent improvement. Both the PPO and DQN agents ultimately converged to optimal returns of 1.0 with no deviation. However, the overall good performance of all methods demonstrates this environment's easy optimisation landscape.
In ComplexMaze-v0, Bayesian optimisation and random search again demonstrated the weakest performance, achieving the lowest returns at every trial despite continuous improvement, eventually

converging to ≈ 1.41, with significant deviation. This indicates fundamental challenges in navigating this environment's complex hyperparameter search space. Conversely, the DQN agent showed quick improvement, reaching near-peak performance within 4.0 trials and achieving the shared highest final return (1.72). The PPO agent exhibited the quickest and steepest improvement, thanks to poor initial returns; the PPO agent quickly converged to promising configurations. However, due to sensitivity to initial returns, the PPO agent once converged prematurely to poorly performing configurations. Bayesian optimisation and random search showed the highest variance, correlating with their poor overall performance.

All methods produced nearly identical learning curves for Sepsis/ICU-Sepsis-v2 (Figure 5), with returns minimally changing throughout optimisation. The small y-axis scale highlights the absence of meaningful differentiation, consistent with this environment's flat reward surface.

These results show that the convergence speeds of HPO methods depend heavily on the environment when trials are limited. The DQN and PPO agents perform strongly in ComplexMaze-v0 and SimpleMaze-v0, showing they learned transferable policies to propose promising hyperparameters. However, in Sepsis/ICU-Sepsis-v2, the flat reward structure reduced the benefits of all learning-based methods.

6.3 RQ3: Budget Increase from 10 to 50 trials

We compare the methods with 10 and 50-trial budgets on the SimpleMaze-v0, ComplexMaze-v0 and Sepsis/ICU-Sepsis-v2 environments. Figure 7 presents the final mean return ($\pm$ standard deviation) for each method. Rank changes between methods are indicated by asterisks above corresponding bars. In ComplexMaze-v0 (panel a), from the front-runners at 10 trials: DQN (1.721 ± 0.022) and PPO (1.719 ± 0.007) agents, only the PPO drops to third place at 50 trials. Random search improves dramatically, going from a return of 1.412 to a return of 1.740, jumping to second place at the 50-trial budget. However, at 50 trials, the best three methods have very similar final mean returns, making their differences tiny. Bayesian optimisation underperforms at 50 trials (1.591 ± 0.301). Both random search and Bayesian optimisation with a greater budget can explore more, thus significantly improving their performance. The Gaussian process behind Bayesian optimisation is now able to identify some patterns, making it behave differently from random search.

For SimpleMaze-v0, in the 50-trial budget, both baselines now reach the maximum reward of 1.0 in all their runs, as they have had more trials to locate promising configurations. The asterisks indicate their improved performance, as all methods now have the same final mean return. Clearly, this environment is trivial as all methods in the 10-trial budget essentially reached the maximum performance. For Sepsis/ICU-Sepsis-v2 (panel b), all methods maintain statistically indistinguishable performance across trial budgets ($\approx 0.798 \pm 0.003$), indicating no exploitable structure for additional optimisation. Although rank changes occur, practical performance differences are negligible.

The improved performance of all HPO methods in ComplexMaze-v0 with a larger budget reveals that traditional methods can match the performance of reinforcement learning-based hyperparameter optimisation methods. The DQN and PPO agents rapidly propose good hyperparameter configurations but can be surpassed in final mean returns by traditional baselines with larger budgets.

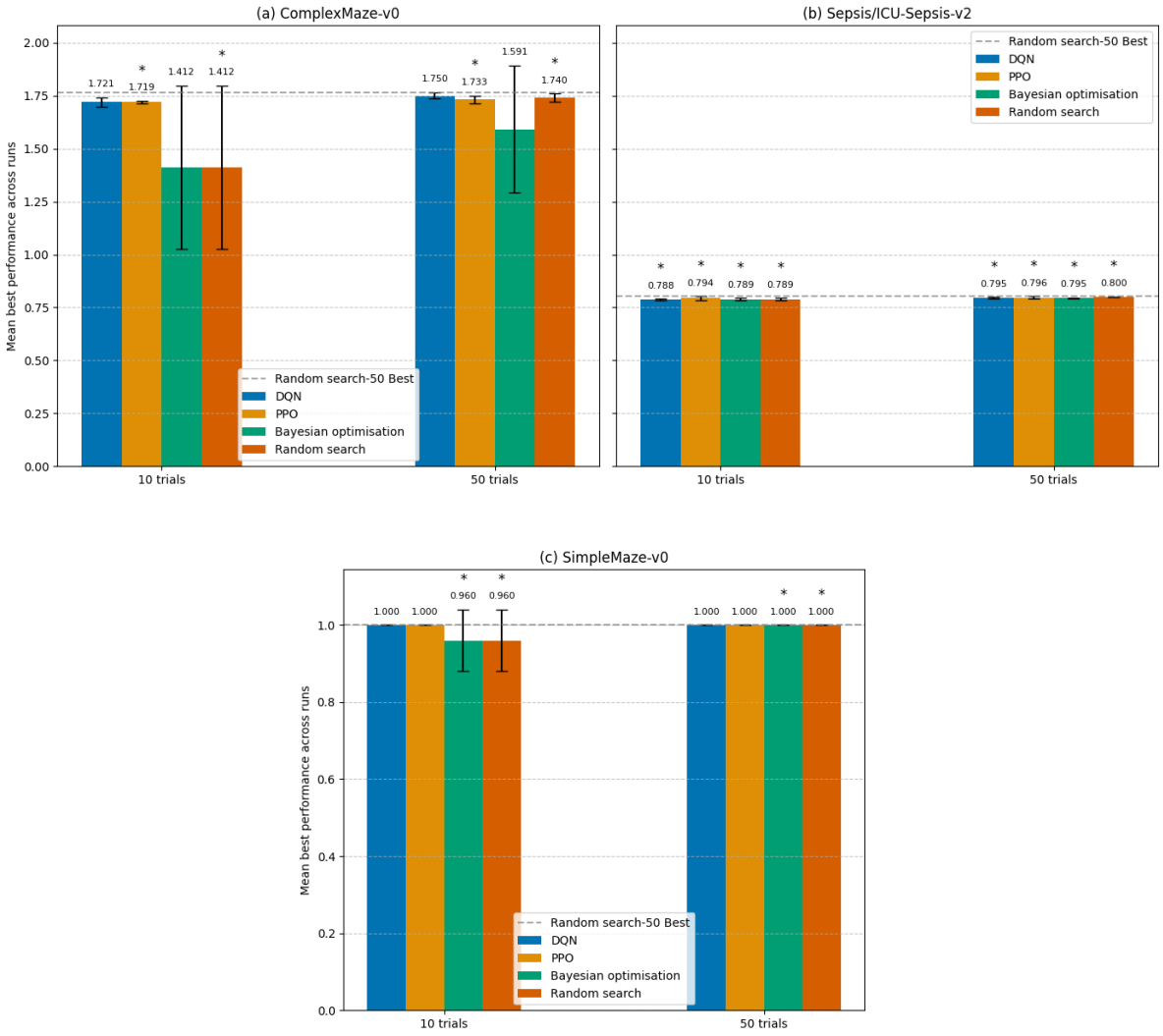


Figure 7: Final mean return over 5 seeds (mean ± SD) at 10 and 50 trial budgets. Panel a) ComplexMaze-v0, panel b) Sepsis/ICU-Sepsis-v2, panel c) SimpleMaze-v0

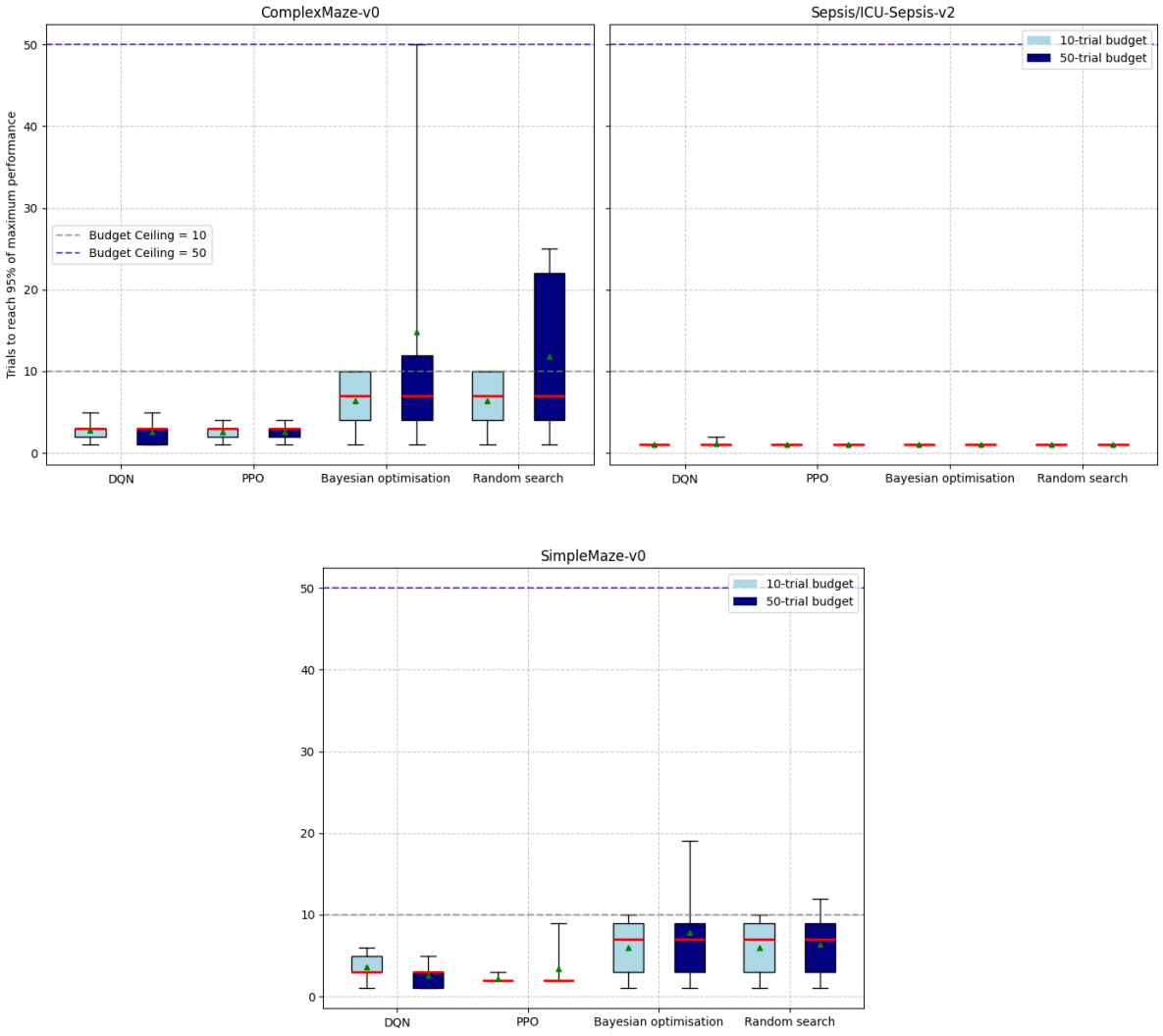


Figure 8: Box-plots of trials required to reach 95% of the best-found return on (a) ComplexMaze-v0, (b) Sepsis/ICU-Sepsis-v2 and (c) SimpleMaze-v0 under 10 and 50 trial budgets. Boxes show median and IQR across five seeds; whiskers extend to min–max. Lower boxes indicate faster convergence.

To quantify convergence speed under increased evaluation budgets, we measured the median trials required to reach 95% of the best-found return. Figure 8 presents these results as comparative box-plots for 10 and 50 trial budgets.

In ComplexMaze-v0 (panel a), median trials remain stable for both the DQN and PPO agents (median = 3.0 across budgets). The baselines, in general, take advantage of the increased budget, with the IQR increasing, but the medians staying at 7.0. The IQR increases from 6.0 to 8.0 for Bayesian optimisation, and increases from 6.0 to 18.0 for random search, indicating poor consistency in locating promising configurations. This confirms previous findings that the DQN and PPO agents maintain the observed superiority even at larger budgets, while Bayesian optimisation and random search demonstrate improved performance with larger budgets.

For SimpleMaze-v0, there are no substantial changes, indicating that this environment is trivial. The superiority of the PPO (median = 3.0) and DQN (median = 2.0) agents remains the same

as in the 10-trial budget. The baselines both hold the same median of 7.0 and IQR of 6.0 across budgets. Bayesian optimisation has a smaller range of values than random search at the larger budget, showing that the Gaussian process has had enough data to form a useful surrogate model. For Sepsis/ICU-Sepsis-v2 (panel b), all methods achieve the 95% threshold within a median of 1.0 trial (IQR = 0.0) under both budgets, confirming additional sampling provides no convergence acceleration on this reward landscape. The singular exception is the one run of the DQN agent requiring 2.0 trials at a budget of 50 trials, which represents sampling noise rather than systematic behaviour.

Random search can perform well at small budgets due to its exploratory nature, but has a higher chance of success with larger budgets. Bayesian optimisation shows improvement, thanks to a more informative Gaussian process, but struggles with the inherently complex hyperparameter space of reinforcement learning.

Figure 9: Complete metrics at the 10-trial budget. Final mean return shows mean ± SD over 5 seeds. Trials to 95% of best return is reported as the median, IQR and SD; Success Rate @ 40% is the proportion of runs reaching 95% of the best return within the first 4 trials; Good-Performing Configurations @ 40% median number of hyperparameter configurations achieving at least 95% of the best return within the first 4 trials; Rank: 1 = best, with ties sharing the same rank.

Environment	**Method**	**Final Mean Return ± SD**	**Median Trials to 95%**	**IQR of Trials to 95%**	**SD of Trials to 95%**	**SR @ 40%**	**GPC @ 40%**	**Rank**
SimpleMaze-v0	Random search	0.960 ± 0.080	7.0	6.0	3.464	0.4	0.0	3
	Bayesian optimisation	0.960 ± 0.080	7.0	6.0	3.464	0.4	0.0	3
	DQN Agent	1.000 ± 0.000	3.0	2.0	1.744	0.6	1.0	2
	PPO Agent	1.000 ± 0.000	2.0	1.0	0.400	1.0	2.0	1
ComplexMaze-v0	Random search	1.412 ± 0.385	7.0	6.0	3.499	0.6	1.0	3
	Bayesian optimisation	1.412 ± 0.385	7.0	6.0	3.499	0.6	1.0	3
	DQN Agent	1.721 ± 0.022	3.0	1.0	1.327	0.8	1.0	1
	PPO Agent	1.719 ± 0.007	3.0	1.0	1.020	1.0	1.0	2
Sepsis/ICU-Sepsis-v2	Random search	0.789 ± 0.005	1.0	0.0	0.0	1.0	4.0	2
	Bayesian optimisation	0.789 ± 0.005	1.0	0.0	0.0	1.0	4.0	2
	DQN Agent	0.788 ± 0.003	1.0	0.0	0.0	1.0	4.0	3
	PPO Agent	0.794 ± 0.009	1.0	0.0	0.0	1.0	4.0	1

Table 9 summarises key performance metrics at the 10-trial budget for ComplexMaze-v0 and

Sepsis/ICU-Sepsis-v2.

In ComplexMaze-v0, the PPO agent demonstrates the quickest ability to find good configurations and perfect early-stage performance (SR@40% = 1.0). DQN achieves a marginally higher final mean return but with worse early-stage performance (SR@40% = 0.8). Bayesian optimisation and random search consistently perform poorly on convergence metrics (median = 7.0 trials, SR@40% = 0.6). All the methods have the same GPC@40% scores of 1.0. However, the reinforcement learning-based hyperparameter optimisation methods still perform better in this metric with more context. The mean for this metric (mean number of promising configurations in 4 trials) is 1.2 for the DQN agent, and 1.6 for the PPO agent, compared to a mean of 0.8 for both baselines.

In the SimpleMaze-v0 environment, one can clearly see the superiority of both the DQN and PPO agents, as their SR@40 (0.6 and 1.0, respectively) and GPC@40 (1.0 and 2.0, respectively) are both better than those of the baselines (both SR@40 = 0.4 and GPC@40 = 0.0). The reinforcement learning-based hyperparameter optimisation methods can locate promising configurations more quickly and consistently. For Sepsis/ICU-Sepsis-v2, all methods show statistically indistinguishable performance. There are universally good convergence metrics (median = 1.0 trial) and perfect success rates (SR@40% = 1.0). Moreover, the statistically insignificant differences in final mean returns further show that the methods are all roughly equal in terms of performance for this environment, and that the ranks for this environment can be ignored.

Together, these results establish a performance baseline, providing reference rankings for extended budget analysis. The DQN and PPO agents' low variances and strong early performance show their ability to quickly suggest reliable hyperparameters, likely due to effective learning from the offline training data.

Figure 10: Complete metrics at the 50-trial budget (40% corresponds to 20 trials now), same metrics as in the above (Table 9)

Environment	Method	Final Mean Return ± SD	Median Trials to 95%	IQR of Trials to 95%	SD of Trials to 95%	SR @ 40%	GPC @ 40%	Rank
SimpleMaze-v0	Random search	1.000 ± 0.000	7.0	6.0	3.980	1.0	3.0	4
	Bayesian optimisation	1.000 ± 0.000	7.0	6.0	6.274	1.0	6.0	3
	DQN Agent	1.000 ± 0.000	3.0	2.0	1.497	1.0	5.0	2
	PPO Agent	1.000 ± 0.000	2.0	0.0	2.800	1.0	9.0	1
ComplexMaze-v0	Random search	1.740 ± 0.020	7.0	18.0	9.786	0.6	1.0	3
	Bayesian optimisation	1.591 ± 0.301	7.0	8.0	17.971	1.0	3.0	4
	DQN Agent	1.750 ± 0.015	3.0	2.0	1.497	1.0	6.0	1
	PPO Agent	1.733 ± 0.018	3.0	1.0	1.020	1.0	8.0	2
Sepsis/ICU-Sepsis-v2	Random search	0.800 ± 0.001	1.0	0.0	0.0	1.0	20.0	1
	Bayesian optimisation	0.795 ± 0.002	1.0	0.0	0.0	1.0	20.0	3
	DQN Agent	0.795 ± 0.003	1.0	0.0	0.400	1.0	20.0	4
	PPO Agent	0.796 ± 0.006	1.0	0.0	0.0	1.0	20.0	2

As can be seen in Table 10 in ComplexMaze-v0, the DQN agent achieves the highest final mean return (1.750 ± 0.015), with good convergence metrics (SR@40% = 1.0, GPC@40% = 6.0, median = 3.0). The PPO agent has a slightly lower final mean return (1.733 ± 0.018), with marginally better convergence metrics (SR@40% = 1.0, GPC@40% = 8.0, median = 3.0). Random search achieves a final mean return similar to the other two methods (1.740 ± 0.020), with far worse convergence metrics (SR@40% = 0.6, GPC@40% = 1.0, median = 7.0), similar to the 10-trial budget. Bayesian optimisation maintains the worst final mean return (1.591 ± 0.301), with slightly better convergence metrics compared to random search (SR@40% = 1.0, GPC@40% = 3.0, median = 7.0). Both the DQN and PPO agents have superior convergence metrics compared to the baselines, reaching the threshold more quickly and consistently. Bayesian optimisation's convergence metrics improve with the increased budget, as the Gaussian process can use more trials to identify patterns. Random search shows improved final return, yet remains outperformed by the DQN agent. The PPO agent emerges as the most effective method for finding promising configurations quickly, further supported by its IQR of 1.0, while DQN still performs well, with a respectable IQR of 2.0. The increased budget illustrates a clear trade-off: baselines can yield equal or even marginally better performance with many trials, whereas reinforcement learning-based hyperparameter optimisation methods require few trials to achieve comparable performance.
In SimpleMaze-v0, we can see that the reinforcement learning-based hyperparameter optimisation methods still maintain their superiority. However, Bayesian optimisation actually overtakes the DQN agent in GPC@40, as it achieves 6.0, compared to the DQN's 5.0. This is the result of the Bayesian optimisation's more informative Gaussian process. In the first 10 trials, Bayesian optimisation behaves identically to random search, to gather information and identify patterns, which is easier thanks to the trivial structure of SimpleMaze-v0. After these 10 trials, it can exploit the Gaussian process to select more promising configurations, whereas the DQN continues to explore.
For Sepsis/ICU-Sepsis-v2, all methods consistently achieve the 95% threshold within a median of 1.0 trial (IQR = 0.0), confirming that additional budget reveals no exploitable structure. Clearly, the optimal HPO method hinges on environment characteristics, as the flat reward structure of Sepsis/ICU-Sepsis-v2 mitigates the advantages of all methods.

7 Discussion

This chapter interprets the experimental findings in relation to the three research sub-questions, situates them within the context of prior work, and discusses their implications and limitations. Overall, the results suggest that the effectiveness of reinforcement-learning-based hyperparameter optimisation methods depends strongly on both the budget and the structure of the test environment.

7.1 RQ1: Reproduction

Across three unseen environments, the DQN agent quickly converged to high-return configurations in SimpleMaze-v0 and ComplexMaze-v0, reaching 95% of the best return within a few trials (e.g. ComplexMaze-v0: median = 3). Increasing rewards, shorter episode lengths, and decreasing TD errors confirm successful learning [SB98, KLM96] and indicate that the optimisation policy progressively selects effective hyperparameter configurations [JGS19]. This demonstrates that Hyp-RL transfers successfully to tabular reinforcement learning tasks.

Performance was strongly environment-dependent. In SimpleMaze-v0, a ceiling effect occurred, with all methods quickly reaching maximum performance. In ComplexMaze-v0, the DQN agent exhibited quick and stable convergence, whereas random search and Bayesian optimisation produced highly variable results. In Sepsis/ICU-Sepsis-v2, the flat reward surface led to indistinguishable outcomes across methods, consistent with the difficulty of optimising noisy or sparse rewards [SB98, KLM96]. These findings suggest that transfer is most effective for tasks that are similar to the training data. However, the experimental setup may overestimate transferability to more complex domains, a limitation also noted in the AutoML survey, where generalisation and robust evaluations remain open challenges [PRS+22]

7.2 RQ2: 10-Trial Performance Comparison

Under a 10-trial budget, the relative advantage of reinforcement learning-based optimisers became evident in structured environments. In ComplexMaze-v0, the DQN and PPO agents achieved similar high mean returns with low variance, outperforming Bayesian optimisation and random search. This supports prior evidence that reinforcement learning-based hyperparameter optimisation can efficiently reuse prior experience under limited evaluation budgets [JGS19] , and that such quick convergence is vital in practical hyperparameter tuning scenarios [ELR23]. Both Bayesian optimisation and random search showed high variability and low mean returns, as they both were inefficient and unstable. This is a result of Bayesian optimisations' sensitivity to low budgets [SSW+16, SLA12].
In Sepsis/ICU-Sepsis-v2 and SimpleMaze-v0, performance was nearly identical across methods due to flat or trivial reward structures. Thus, the benefit of reinforcement learning-based hyperparameter optimisation methods lies in the objective function having an exploitable structure or being complex.

7.3 RQ3: Budget Increase from 10 to 50 trials

With a larger 50-trial budget, performance rankings shifted. In ComplexMaze-v0, both random search and Bayesian optimisation improved substantially, with random search surpassing the PPO agents in final mean return. The PPO agent, however, maintained the fastest early convergence. This illustrates a trade-off noted in the literature: model-based HPO methods (Bayesian optimisation) can eventually catch up or outperform learning-based HPO methods (PPO, DQN) as evaluation data accumulates [SSW+16, HKV19]. For SimpleMaze-v0, the performance of Bayesian optimisation improved with the 50-trial budget, as it was more consistent in choosing promising configurations. This is due to the simplicity of SimpleMaze-v0, allowing the Gaussian process to identify exploitable patterns, unlike in the stochastic ComplexMaze-v0. In Sepsis/ICU-Sepsis-v2, performance remained similar across methods, confirming that the flat reward landscape limits the potential of adaptive search.

7.4 Limitations

Several limitations qualify the conclusions drawn from this work:

- **Baselines** Random search, and Bayesian optimisation were selected as representative yet relatively simple baselines. More recent multi-fidelity or hybrid methods, such as Hyperband [LJD+17] or BOHB [PRS+22], may provide a more up-to-date and accurate comparison.

- **Generalisability** The search space for the reinforcement learning-based hyperparameter optimisation methods was deliberately restricted to a discrete two-dimensional domain to allow controlled evaluation and ease of interpretability. Consequently, performance in higher-dimensional or more complex search spaces remains to be validated.
- **Resource cost** Training the optimisation agent and constructing the offline dataset are computationally expensive, as noted in other AutoRL studies [PRS+22]. The benefit of reinforcement learning-based hyperparameter optimisation methods must outweigh these costs to justify practical adoption.
- **Recursive HPO** The optimisation agents themselves introduce additional hyperparameters, raising the possibility of recursive tuning. Developing approaches for such self-optimising systems remains an open research question [PRS+22].

Despite these limitations, the study demonstrates that reinforcement learning-based hyperparameter optimisation methods can be competitive under low budgets and that they can be successfully transferred to new domains.

8 Conclusion and Future Work

This thesis evaluated whether the Hyp-RL framework, originally developed for tuning neural networks, can be effectively transferred to tune SARSA agents under limited evaluation budgets. The results demonstrate that the framework retains its quick convergence, but also highlight the conditions under which traditional hyperparameter optimisation methods remain competitive.
RQ1 investigated whether the original DQN agent from the Hyp-RL framework can be successfully applied to tabular reinforcement learning tasks. The results indicate that the DQN agent effectively adapts to this new setting, outperforming random search and Bayesian optimisation by reaching 95% of the best observed return in only three trials and achieving the lowest performance variance on ComplexMaze-v0. This reproduces similar findings originally reported for neural network tuning [JGS19].
RQ2 examined how these methods perform under low evaluation budgets of 10 trials. In this setting, both the DQN and PPO optimisers outperformed traditional baselines on SimpleMaze-v0 and ComplexMaze-v0. The performance advantage largely disappeared in noisier environments, such as Sepsis/ICU-Sepsis-v2, suggesting that the benefits of reinforcement learning-based hyperparameter methods may depend on task complexity and signal stability.
RQ3 explored performance under more generous evaluation budgets of 50 trials. On ComplexMaze-v0 and SimpleMaze-v0, the DQN and PPO agents maintained faster convergence than random search and Bayesian optimisation. In contrast, on Sepsis/ICU-Sepsis-v2, all methods exhibited low and statistically indistinguishable performance, indicating that environment stochasticity and reward noise can limit the effectiveness of adaptive optimisation strategies.
Taken together, these findings clarify where reinforcement learning-based hyperparameter optimisation methods are best, under low evaluation budgets and in moderately complex environments. Here, the methods yield measurable improvements in convergence speed and performance stability. When evaluation budgets are larger or the problem is noisy, traditional methods such as Bayesian optimisation remain more reliable.

Future Work The findings of this thesis suggest several avenues for further research:

- **Mitigate Q-value overestimation:** Currently, the DQN agent exhibits the well-known overestimation bias of standard Q-learning. Implementing a Double DQN variant, where action selection and evaluation are separated, could address this issue. Benchmarking the new Double DQN against the current agent would show whether the optimising agent can learn faster and improve performance [SB98].
- **Develop robust meta-features for reinforcement learning environments:** The meta-features used in this work were designed in an ad-hoc manner due to limited prior literature on meta-features specific to reinforcement learning. Future work should systematically identify features that capture environment characteristics and evaluate how the predictive power of these features translates into improved HPO performance.
- **Expand the hyperparameter space:** The SARSA agent has three primary hyperparameters: the learning rate, exploration rate, and discount factor. In this work, the discount factor was fixed to 0.99 to reduce the search space to two dimensions. Expanding the hyperparameter search space to include the discount factor and other relevant parameters would expose reinforcement learning-based hyperparameter optimisation methods to more sensitive reward landscapes and test their scalability.
- **Apply to deep reinforcement learning:** While this thesis argued that applying the proposed reinforcement learning-based hyperparameter optimisation to tabular agents may be unnecessarily complex, extending these methods to deep reinforcement learning algorithms would provide a more rigorous test of their effectiveness. Deep reinforcement learning involves larger hyperparameter spaces and highly non-stationary reward surfaces [SB98], offering a more challenging domain in which the proposed methods may better align with the underlying task complexity.
- **Beyond traditional baselines:** The proposed methods were benchmarked against random search and Bayesian optimisation. Future work should evaluate their generalisability against more advanced methods, such as model-based and population-based optimisation approaches.

References

[BB12] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[BGNR17] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing Neural Network Architectures using Reinforcement Learning. In *International Conference on Learning Representations*, 2017.

[BVL+25] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa M. Zintgraf, Chelsea Finn, and Shimon Whiteson. A Tutorial on Meta-Reinforcement Learning. *Foundations and Trends Machine Learning*, 18(2-3):224–384, 2025.

[BWL+24] Mitra Baratchi, Can Wang, Steffen Limmer, Jan N. van Rijn, Holger H. Hoos, Thomas Bäck, and Markus Olhofer. Automated machine learning: past, present and future. *Artificial Intelligence Review*, 57(5):122, 2024.

[CGT24] Kartik Choudhary, Dhawal Gupta, and Philip S. Thomas. ICU-Sepsis: A Benchmark MDP Built from Real Medical Data. *Reinforcement Learning Journal*, 4:1546–1566, 2024.

[CHHS20] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *International Conference on Machine Learning*, pages 2048–2056, 2020.

[ELR23] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in Reinforcement Learning and How To Tune Them. In *International Conference on Machine Learning*, pages 9104–9149, 2023.

[FFR+21] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[HAMS22] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022.

[HIB+18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3207–3214, 2018.

[HKV19] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning - Methods, Systems, Challenges*. Springer, 2019.

[JGS19] Hadi S. Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. Hyp-RL : Hyperparameter Optimization by Reinforcement Learning. *CoRR*, abs/1906.11527, 2019.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[LJD+17] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18:185:1–185:52, 2017.

[PRS+22] Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, Frank Hutter, and Marius Lindauer. Automated Reinforcement Learning (AutoRL): A Survey and Open Problems. *Journal of Artificial Intelligence Research*, 74:517–568, 2022.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - An Introduction*. MIT Press, 1998.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, pages 2960–2968, 2012.

[SSW+16] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

[TKT+24] Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, abs/2407.17032, 2024.

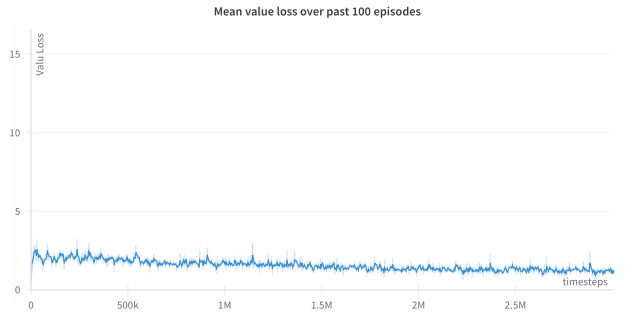
[WLC23] Jia Wu, Xiyuan Liu, and Senpeng Chen. Hyperparameter optimization through context-based meta-reinforcement learning with task-aware representation. *Knowledge Based Systems*, 260:110160, 2023.

[ZL16] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.

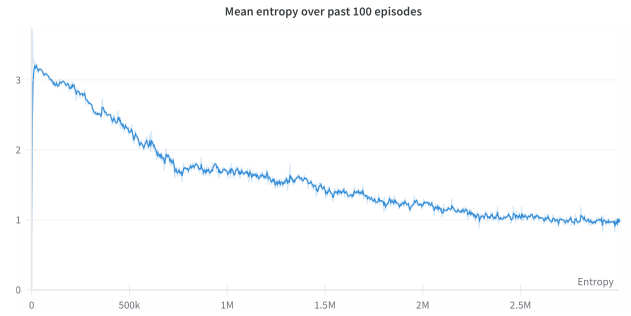
[ZTL+25] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground. *CoRR*, abs/2502.08844, 2025.

[ZVSL18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

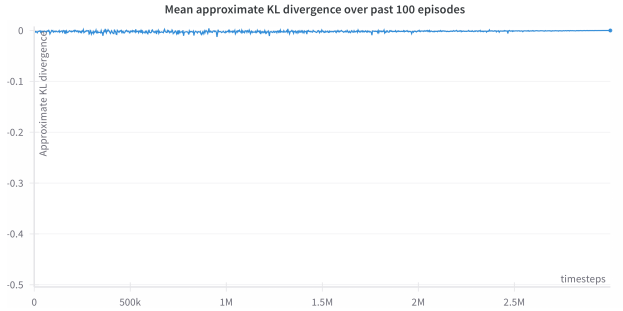
A Additional Results



(a) Mean value loss of the PPO agent, calculated over a rolling window of the previous 100 episodes and smoothed with a factor of 0.5.



(b) Mean entropy of the PPO agent, calculated over a rolling window of the previous 100 episodes and smoothed with a factor of 0.5.



(c) Mean approximate KL divergence of the PPO agent, calculated over a rolling window of the previous 100 episodes and smoothed with a factor of 0.5.

Figure 11: The remainder of the training curves of the PPO agent are designed for tabular SARSA hyperparameter optimisation.

Run	Method	Best Value	Best Configuration
1	Random search	1.022	(0.383, 0.964)
2	Random search	0.868	(0.346, 0.186)
3	Random search	1.704	(0.026, 0.436)
4	Random search	1.757	(0.279, 0.649)
5	Random search	1.710	(0.006, 0.976)
1	Bayesian optimisation	1.220	(0.383, 0.964)
2	Bayesian optimisation	0.868	(0.346, 0.186)
3	Bayesian optimisation	1.704	(0.026, 0.436)
4	Bayesian optimisation	1.757	(0.279, 0.649)
5	Bayesian optimisation	1.710	(0.006, 0.976)
1	DQN Agent	1.760	(0.020, 0.200)
2	DQN Agent	1.696	(0.010, 0.400)
3	DQN Agent	1.712	(0.050, 0.600)
4	DQN Agent	1.728	(0.010, 0.300)
5	DQN Agent	1.710	(0.050, 0.500)
1	PPO Agent	1.723	(0.050, 0.300)
2	PPO Agent	1.724	(0.050, 0.500)
3	PPO Agent	1.712	(0.050, 0.600)
4	PPO Agent	1.728	(0.010, 0.300)
5	PPO Agent	1.710	(0.050, 0.500)

Table 2: Detailed Results for ComplexMaze-v0 for 10 trials, configurations are listed as (learning rate, ϵ)

Run	Method	Best Value	Best Configuration
1	Random search	0.785	(0.646, 0.424)
2	Random search	0.795	(0.878, 0.205)
3	Random search	0.783	(0.514, 0.135)
4	Random search	0.795	(0.511, 0.291)
5	Random search	0.789	(0.715, 0.973)
1	Bayesian optimisation	0.785	(0.646, 0.424)
2	Bayesian optimisation	0.795	(0.878, 0.205)
3	Bayesian optimisation	0.783	(0.514, 0.135)
4	Bayesian optimisation	0.795	(0.511, 0.291)
5	Bayesian optimisation	0.789	(0.715, 0.973)
1	DQN Agent	0.787	(0.050, 0.500)
2	DQN Agent	0.785	(0.010, 0.500)
3	DQN Agent	0.791	(0.020, 0.200)
4	DQN Agent	0.786	(0.050, 0.600)
5	DQN Agent	0.792	(0.050, 0.600)
1	PPO Agent	0.789	(0.500, 0.200)
2	PPO Agent	0.796	(0.500, 0.300)
3	PPO Agent	0.808	(0.050, 0.500)
4	PPO Agent	0.780	(0.100, 0.400)
5	PPO Agent	0.797	(0.500, 0.050)

Table 3: Detailed Results for Sepsis/ICU-Sepsis-v2 for 10 trials, configurations are listed as (learning rate, ϵ)

Run	Method	Best Value	Best Configuration
1	Random search	0.800	(0.383, 0.964)
2	Random search	1.000	(0.559, 0.417)
3	Random search	1.000	(0.026, 0.436)
4	Random search	0.800	(0.279, 0.649)
5	Random search	1.000	(0.216, 0.698)
1	Bayesian optimisation	0.800	(0.383, 0.964)
2	Bayesian optimisation	1.000	(0.559, 0.417)
3	Bayesian optimisation	1.000	(0.026, 0.436)
4	Bayesian optimisation	1.000	(0.279, 0.649)
5	Bayesian optimisation	1.000	(0.216, 0.698)
1	DQN Agent	1.000	(0.050, 0.500)
2	DQN Agent	1.000	(0.010, 0.400)
3	DQN Agent	1.000	(0.010, 0.400)
4	DQN Agent	1.000	(0.050, 0.600)
5	DQN Agent	1.000	(0.050, 0.600)
1	PPO Agent	1.000	(0.100, 0.600)
2	PPO Agent	1.000	(0.300, 0.400)
3	PPO Agent	1.000	(0.300, 0.600)
4	PPO Agent	1.000	(0.050, 0.600)
5	PPO Agent	1.000	(0.100, 0.600)

Table 4: Detailed Results for SimpleMaze-v0 for 10 trials, configurations are listed as (learning rate, ϵ)

Run	Method	Best Value	Best Configuration
1	Random search	0.800	(0.096, 0.838)
2	Random search	0.798	(0.700, 0.589)
3	Random search	0.802	(0.483, 0.640)
4	Random search	0.799	(0.067, 0.597)
5	Random search	0.799	(0.182, 0.498)
1	Bayesian optimisation	0.791	(1.000, 1.000)
2	Bayesian optimisation	0.796	(0.665, 1.000)
3	Bayesian optimisation	0.795	(0.130, 1.000)
4	Bayesian optimisation	0.797	(0.076, 0.871)
5	Bayesian optimisation	0.796	(0.233, 0.806)
1	DQN Agent	0.792	(0.050, 0.300)
2	DQN Agent	0.792	(0.300, 0.300)
3	DQN Agent	0.795	(0.500, 0.100)
4	DQN Agent	0.793	(0.010, 0.600)
5	DQN Agent	0.797	(0.500, 0.050)
1	PPO Agent	0.801	(0.050, 0.300)
2	PPO Agent	0.796	(0.500, 0.300)
3	PPO Agent	0.808	(0.050, 0.500)
4	PPO Agent	0.795	(0.500, 0.600)
5	PPO Agent	0.790	(0.050, 0.400)

Table 5: Detailed Results for Sepsis/ICU-Sepsis-v2 for 50 trials, configurations are listed as (learning rate, ϵ)

Run	Method	Best Value	Best Configuration
1	Random search	1.765	(0.096, 0.837)
2	Random search	1.734	(0.130, 0.288)
3	Random search	1.733	(0.113, 0.973)
4	Random search	1.757	(0.278, 0.649)
5	Random search	1.710	(0.006, 0.976)
1	Bayesian optimisation	1.783	(0.148, 0.926)
2	Bayesian optimisation	0.992	(0.543, 1.000)
3	Bayesian optimisation	1.706	(0.046, 0.458)
4	Bayesian optimisation	1.757	(0.279, 0.649)
5	Bayesian optimisation	1.720	(0.117, 1.000)
1	DQN Agent	1.749	(0.050, 0.400)
2	DQN Agent	1.742	(0.020, 0.200)
3	DQN Agent	1.712	(0.050, 0.600)
4	DQN Agent	1.728	(0.010, 0.300)
5	DQN Agent	1.710	(0.050, 0.500)
1	PPO Agent	1.762	(0.050, 0.400)
2	PPO Agent	1.742	(0.020, 0.200)
3	PPO Agent	1.712	(0.050, 0.600)
4	PPO Agent	1.731	(0.020, 0.400)
5	PPO Agent	1.717	(0.020, 0.300)

Table 6: Detailed Results for ComplexMaze-v0 for 50 trials, configurations are listed as (learning rate, ϵ)

Run	Method	Best Value	Best Configuration
1	Random search	1.000	(0.781, 0.462)
2	Random search	1.000	(0.559, 0.417)
3	Random search	1.000	(0.026, 0.436)
4	Random search	1.000	(0.279, 0.649)
5	Random search	1.000	(0.216, 0.698)
1	Bayesian optimisation	1.000	(0.362, 0.809)
2	Bayesian optimisation	1.000	(0.559, 0.417)
3	Bayesian optimisation	1.000	(0.026, 0.436)
4	Bayesian optimisation	1.000	(0.279, 0.649)
5	Bayesian optimisation	1.000	(0.216, 0.698)
1	DQN Agent	1.000	(0.050, 0.600)
2	DQN Agent	1.000	(0.010, 0.400)
3	DQN Agent	1.000	(0.100, 0.400)
4	DQN Agent	1.000	(0.050, 0.600)
5	DQN Agent	1.000	(0.050, 0.600)
1	PPO Agent	1.000	(0.100, 0.600)
2	PPO Agent	1.000	(0.300, 0.400)
3	PPO Agent	1.000	(0.300, 0.600)
4	PPO Agent	1.000	(0.100, 0.400)
5	PPO Agent	1.000	(0.010, 0.600)

Table 7: Detailed Results for SimpleMaze-v0 for 50 trials, configurations are listed as (learning rate, ϵ)