Optimizing Robot Tutor Strategies via Markov Decision Process Modeling

Vico Winner Sebastian Aritonang[1]

[1]2306219083

Abstract

Personalized education at scale requires autonomous tutoring agents that can balance pedagogical intensity against student fatigue and engagement across realistic, full-day study sessions. This work formalises the Robot Tutor problem as a Markov Decision Process (MDP) (Puterman, 1994; Sutton and Barto, 2018) over a 1440-state space that captures a 24-hour clock, a ten-level proficiency variable, a three-level fatigue variable, and a binary engagement variable, and seeks an optimal tutoring policy via deep reinforcement learning. Four deep RL algorithms are evaluated on the `RobotTutor-v2` Gymnasium environment (Towers et al., 2024) across four independent random seeds: Deep Q-Network (DQN) (Mnih et al., 2015), Proximal Policy Optimization (PPO) (Schulman et al., 2017), Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and Soft Actor-Critic (SAC) (Haarnoja et al., 2018), the last trained through a continuous-action wrapper. Soft Actor-Critic attains the best performance, the highest reward (826.8 ± 17.0 in a held-out evaluation of 200 episodes per seed) and a 100% expert-proficiency reach rate, followed by DQN (757.7) and TRPO (740.7), while PPO is the least stable (309.1 ± 436.8) and reaches expert in only 68.9% of episodes. SAC converges in roughly 236 episodes and keeps high-fatigue exposure low, making it the recommended algorithm for this problem class; all four learned policies far surpass random-action and fixed-schedule baselines. Classical dynamic programming baselines and tabular RL methods are reported in the Appendix.

1 Introduction

1.1 Problem Description

Personalized education is widely regarded as the gold standard in pedagogy, offering instruction tailored to a student's pace, prior knowledge, and current state. Scaling such mentorship to every learner, however, is bottlenecked by the scarcity of expert human educators. Intelligent Tutoring Systems (ITS) and Robot Tutors aim to close this gap by autonomously sequencing pedagogical actions in a way that maximises long-term learning outcomes.

The core difficulty is sequential: a tutoring agent must repeatedly decide *what* to do next (lecture, drill, low-intensity practice, game, or rest) while accounting for the student's accumulated fatigue, current proficiency, engagement level, and the time of day. High intensity actions yield faster proficiency gains but accelerate fatigue, and a fatigued student learns much less from the same action. Over a full 24-hour cycle, an effective Robot Tutor must therefore behave anticipatorily rather than greedily, trading immediate learning gains against long-horizon mental cost. Figure 1 illustrates this interaction.

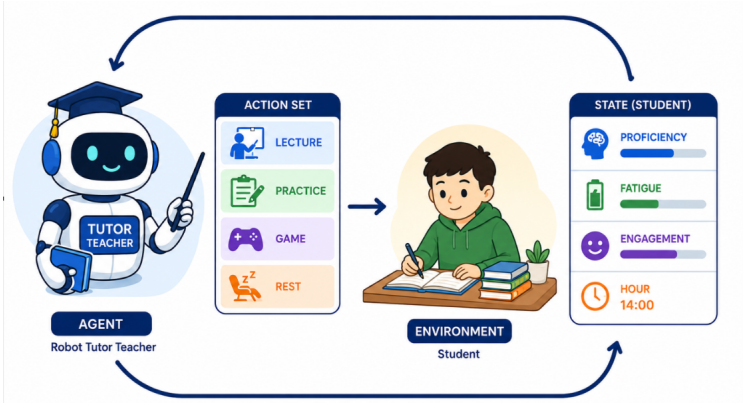

Figure 1: Illustration of the Robot Tutor problem: the tutoring agent sequences pedagogical actions in response to the student's evolving proficiency, fatigue, and engagement over a daily study cycle.

1.2 Motivation and Context

The Robot Tutor problem sits at the intersection of two longstanding challenges in education technology: *adaptive content sequencing* and *student state modeling*. Human tutors naturally adjust lesson difficulty, pacing, and delivery style in response to subtle cues about a student's engagement and mental fatigue, yet most existing automated systems rely on fixed curricula or simple heuristics that cannot capture this dynamic interplay. Recent empirical studies have demonstrated the value of moving beyond static sequencing: Ruan et al. (2024) show that a reinforcement learning tutor that adapts its hint-giving strategy to individual students produces the largest learning gains specifically for lower-performing students, a population that fixed curricula systematically underserve. Similarly, Chung et al. (2026) find that an adaptive problem-selection agent, driven by student-chatbot interaction signals, improves unassisted exam performance by 0.15 standard deviations over a five-month real-world deployment in ten high schools. Earlier, Bassen et al. (2020) trained a deep RL agent (PPO) to schedule educational activities in real time for an online course, achieving comparable learning gains with substantially fewer activities and lower dropout than fixed or self-directed sequencing.

These three systems establish reinforcement learning as an effective tutoring-policy mechanism, yet each captures only a narrow slice of the student state, none jointly models mental fatigue and engagement, and none releases a reusable open-source environment for controlled algorithm comparison. Table 1 positions our formulation against them along the axes of state/action modelling, learning algorithm, open-source availability, and evaluation. In contrast to prior work, our MDP treats both *fatigue* and *engagement* as first-class state variables over a full 24-hour scheduling horizon, and we release a reproducible Gymnasium environment that enables the head-to-head comparison of four deep RL algorithms.

What these systems share is the need to model the student as a dynamic entity whose *cognitive state* evolves in response to pedagogical actions. Fatigue is a particularly important and undermodeled dimension of that state: in a real deployment, a tutoring system cannot directly observe a student's cognitive load but must infer it from proxies such as response latency, error patterns, and interaction frequency. A Robot Tutor that ignores fatigue will tend to over-schedule high-intensity activities, accelerating cognitive depletion and reducing the effectiveness of later sessions. Engagement compounds this challenge: students who are actively engaged with a learning activity retain information more effectively, yet engagement is transient and action-dependent. Capturing both fatigue and engagement within a principled sequential decision-making framework is the central motivation for our MDP formulation.

We formalise this as a Markov Decision Process (Puterman, 1994) and solve it with deep reinforcement learning (Sutton and Barto, 2018), concentrating on four algorithms: **Deep Q-Network (DQN)** (Mnih et al., 2015), **Proximal Policy Optimization (PPO)** (Schulman et al., 2017), **Trust Region Policy Optimization (TRPO)** (Schulman et al., 2015), and **Soft Actor-Critic (SAC)** (Haarnoja et al., 2018).

Table 1: Comparison of reinforcement-learning tutoring/sequencing systems. "Open?" indicates whether the environment or code is released open-source. Sizes are reported where a discrete count is defined; continuous feature-vector states have no fixed cardinality.

Work	State (size)	Action (count)	RL method(s)	Open?	Evaluation
Ruan et al. (2024)	Learner-feature vector (continuous)	Feedback / hint type (4)	PPO; offline batch RL	No	Real children (269+37), math
Chung et al. (2026)	Student–chatbot interaction signals	Next problem / difficulty	LLM-guided RL	No	Real RCT, 10 schools (5 mo.)
Bassen et al. (2020)	Learner trace: pre-test + completions + grades ($\sim$30-d, cont.)	Next activity (13, incl. post-test)	PPO (actor–critic)	No	Sim. + real RCT (1,987)
This work	(h, k, f, e)**: time, proficiency, fatigue, engagement (1440, discrete)**	Lecture / practice / game / rest (9)	DQN, PPO, TRPO, SAC	**Yes**	Simulation (MDP), 4 seeds

1.3 Contributions

The contribution of this paper is threefold:

1. **MDP Modeling for the Robot Tutor Problem.** We construct a realistic 1440-state Robot Tutor MDP over a 24-hour clock, a ten-level proficiency variable, a three-level fatigue variable, and a binary engagement variable. The engagement state, driven by research on game-based learning (Corbett and Anderson, 1994), captures motivational dynamics: games and active practice raise engagement, and an engaged student benefits from a multiplicative success bonus. Transition probabilities are grounded in Bayesian Knowledge Tracing literature (Corbett and Anderson, 1994), and the reward function is designed to simultaneously incentivise learning, discourage fatigue, and penalise nocturnal study violations.
2. **Gymnasium Implementation of the MDP.** All experiments are conducted through a single shared `RobotTutor-v2` Gymnasium environment (Towers et al., 2024) that exposes the full 1440-state MDP through the standard `reset`/`step` API. The environment is compatible with the Stable-Baselines3 training library, enabling reproducible, seed-controlled comparisons.
3. **Comparison of Deep RL Methods.** We train DQN, PPO, TRPO, and SAC on the same environment with four independent random seeds each, and evaluate them not only on final mean reward but also on convergence speed, training stability, steps to first expert proficiency, and accumulated high-fatigue exposure. Because SAC is defined for continuous action spaces, it is trained through a thin wrapper that exposes the nine discrete actions as a continuous vector and selects the executed action by argmax (Section 3.4). Classical dynamic programming and tabular baselines are reported in the Appendix as a sanity check on the planning optimum.

2 MDP Modelling

The educational interaction between the Robot Tutor and the student is formally modeled as a Markov Decision Process (MDP) (Puterman, 1994; Sutton and Barto, 2018). The framework is defined by the tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathring{p}, p, \mathcal{R} \rangle, \tag{1}$$

Table 2: Components of the State Set $\mathcal{S}$.

Variable	Domain	Description
h (Hour)	$\{0, 1, \ldots, 23\}$	Wall-clock hour within a 24-hour day. Hours $h \in \{22, 23, 0, 1, 2, 3, 4, 5\}$ form the sleep window.
k (Proficiency)	$\{0, 1, \ldots, 9\}$	Student's mastery level, from Novice (0) to Expert (9).
f (Fatigue)	$\{0, 1, 2\}$	Graded mental state: Fresh (0), Moderately tired (1), Highly tired (2).
e (Engagement)	$\{0, 1\}$	Student engagement level: Disengaged (0) or Engaged (1).

where $\mathcal{S}$ represents the state set, $\mathcal{A}$ the action set, $\mathcal{T}$ the time horizon, $\overset{\circ}{p}$ the initial state distribution, p the state transition probability, and $\mathcal{R}$ the reward function.

2.1 State Set $\mathcal{S}$

We conceptualise the learning environment through a four-dimensional state representation that captures, respectively, time progression, knowledge acquisition, mental exhaustion, and student engagement:

$$s = (h, k, f, e). \tag{2}$$

The complete state set is the Cartesian product of these features,

$$\mathcal{S} = \{0, \ldots, 23\} \times \{0, \ldots, 9\} \times \{0, 1, 2\} \times \{0, 1\}, \tag{3}$$

yielding $|\mathcal{S}| = 24 \times 10 \times 3 \times 2 = 1440$ distinct states.

2.2 Action Set $\mathcal{A}$

At every timestep, the agent selects an action from a state-independent discrete set of 9 pedagogical actions:

$$\mathcal{A} = \left\{ \begin{array}{l} \text{Lecture-Theoretical}, \\ \text{Lecture-Practical}, \\ \text{Game-Hard}, \\ \text{Game-Medium}, \\ \text{Game-Easy}, \\ \text{Practice-Hard}, \\ \text{Practice-Medium}, \\ \text{Practice-Easy}, \\ \text{Rest} \end{array} \right\}. \tag{4}$$

2.3 Time Set $\mathcal{T}$

The interaction operates over discrete hourly steps,

$$\mathcal{T} = \{0, 1, 2, \ldots\}, \qquad \Delta t = 1 \text{ hour}. \tag{5}$$

The clock advances modulo 24, cycling between an active learning window ($h \in \{6, \ldots, 21\}$) and a sleep window ($h \in \{22, 23, 0, 1, \ldots, 5\}$). One daily reset event fires when the agent chooses *Rest* at $h = 5$. Note that the agent may choose non-resting action even at sleep windows.

We model the interaction as both *episodic* and *continuing*. In the episodic interpretation, the task has a natural terminal absorbing state when the student reaches expert proficiency ($k = 9$); the

Table 3: Pedagogical Action Definitions.

Action	Pedagogical Description
Lecture-Theoretical	Passive exposition of concepts; moderate learning gain, low cognitive load, and minimal engagement boost.
Lecture-Practical	Interactive lecture with worked examples; highest base proficiency gain, moderate cognitive load, moderate engagement.
Game-Hard	Challenging gamified activity; highest cognitive load and engagement boost, lower success rate due to high difficulty.
Game-Medium	Balanced game at optimal difficulty; highest engagement boost, moderate cognitive load and learning gain.
Game-Easy	Low-difficulty game; moderate engagement, low fatigue, lower learning gain.
Practice-Hard	Rigorous exercise set; highest cognitive load, moderate-to-high learning gain.
Practice-Medium	Standard practice problems; balanced cognitive load and learning gain.
Practice-Easy	Light exercise set; low cognitive load, modest learning gain.
Rest	Recovery action; deterministically resets fatigue to $f' = 0$, no proficiency gain, drops engagement.

infinite-horizon discounted formulation with $\gamma < 1$ handles this gracefully, as future rewards beyond the terminal state contribute zero value. In the continuing interpretation, episodes are repeated cyclically to account for instruction across multiple subjects or learning objectives, allowing the agent to generalise its pedagogical strategy to new students and curricula.

2.4 Initial State Distribution $\overset{\circ}{p}$

Each episode initialises at the first hour of the active window with a well-rested, novice, disengaged student. The initial-state distribution is a Dirac delta centred at $s_0 = (h{=}6,\ k{=}0,\ f{=}0,\ e{=}0)$:

$$\overset{\circ}{p}(s) = \mathbb{I}[h = 6] \cdot \mathbb{I}[k = 0] \cdot \mathbb{I}[f = 0] \cdot \mathbb{I}[e = 0], \qquad \forall\, s \in \mathcal{S}. \tag{6}$$

2.5 Transition Distribution p

We formulate the transition probability $p(s' \mid s, a)$ where $s = (h, k, f, e)$ and $s' = (h', k', f', e')$. The dynamics decompose into (i) the daily overnight reset, (ii) deterministic recovery under ***Rest***, and (iii) stochastic teaching dynamics. In all cases, the hour advances modulo 24:

$$p(h' = (h + 1) \bmod 24 \mid s, a) = 1. \tag{7}$$

(1) Overnight reset (*Rest* at $h = 5$). The overnight reset models the cognitive science insight that sleep simultaneously restores mental capacity and partially consolidates memories (Sutton and Barto, 2018). When the agent chooses ***Rest*** at $h = 5$, the clock advances directly to $h' = 6$, fatigue is deterministically cleared to $f' = 0$, engagement drops to $e' = 0$, and a $50/50$ memory-decay event is applied to proficiency: the student either retains the day's gains ($k' = k$) or loses one level ($k' = k-1$). This stochastic decay represents the imperfect consolidation of learning during sleep: a student who crammed under fatigue is more likely to forget than one who paced their study sessions. Choosing any non-***Rest*** action at $h = 5$ does *not* trigger the reset; the night-violation penalty applies instead.

Table 4: Per-action base success probability $P_{\text{base}}(a)$, fatigue increase probability $P_{\text{fatigue}}(a)$, and engagement increase probability $P_{\text{engage}}(a)$.

Action	$P_{\text{base}}(a)$	$P_{\text{fatigue}}(a)$	$P_{\text{engage}}(a)$
Lecture-Theoretical	0.60	0.30	0.10
Lecture-Practical	0.70	0.40	0.20
Game-Hard	0.40	0.60	0.50
Game-Medium	0.50	0.50	0.60
Game-Easy	0.35	0.30	0.40
Practice-Hard	0.45	0.70	0.25
Practice-Medium	0.55	0.50	0.30
Practice-Easy	0.40	0.30	0.15

$$
\begin{aligned}
p(h' = 6 \mid (5, k, f, e), \text{Rest}) &= 1, \\
p(f' = 0 \mid (5, k, f, e), \text{Rest}) &= 1, \\
p(e' = 0 \mid (5, k, f, e), \text{Rest}) &= 1, \\
p(k' \mid (5, k, f, e), \text{Rest}) &= \begin{cases} 0.5, & k' = \max(0, k-1), \\ 0.5, & k' = k. \end{cases}
\end{aligned} \tag{8}
$$

(2) *Rest* (outside the overnight reset). *Rest* at any hour $h \neq 5$ deterministically clears fatigue, preserves proficiency, and drops engagement:

$$
p(k' = k \mid s, \text{Rest}) = 1, \qquad p(f' = 0 \mid s, \text{Rest}) = 1, \qquad p(e' = 0 \mid s, \text{Rest}) = 1. \tag{9}
$$

(3) Teaching actions. For teaching actions $a \in \mathcal{A} \setminus \{\text{Rest}\}$, three sub-processes evolve independently.

Base success probabilities. Each action is assigned a base proficiency-gain probability $P_{\text{base}}(a)$, derived from Bayesian Knowledge Tracing learning rates (Corbett and Anderson, 1994). The effective success probability attenuates with fatigue and receives a bonus when the student is engaged:

$$
\rho_{\text{succ}}(s, a) = \min(1,\ P_{\text{base}}(a) \cdot \alpha_f + e \cdot \delta_e)\,, \qquad \alpha_0 = 1.0,\ \alpha_1 = 0.2,\ \alpha_2 = 0.1,\ \delta_e = 0.2, \tag{10}
$$

where α_f is the fatigue scale factor and $\delta_e = 0.2$ is the engagement bonus. Proficiency evolves as

$$
p(k' \mid s, a) = \begin{cases} \rho_{\text{succ}}(s, a), & k' = \min(k+1, 9), \\ 1 - \rho_{\text{succ}}(s, a), & k' = k. \end{cases} \tag{11}
$$

Fatigue dynamics. Fatigue escalates independently with action-specific probability $P_{\text{fatigue}}(a)$ (Table 4), capturing cognitive load theory: harder tasks impose higher cognitive load (Sutton and Barto, 2018). Fatigue saturates at $f = 2$:

$$
p(f' \mid s, a) = \begin{cases} P_{\text{fatigue}}(a), & f' = \min(f+1, 2) \quad \text{(escalation)}, \\ 1 - P_{\text{fatigue}}(a), & f' = f \qquad\qquad\ \ \text{(no escalation)}, \end{cases} \quad f \in \{0, 1\}; \tag{12}
$$

for $f = 2$: $p(f' = 2 \mid s, a) = 1$.

Engagement dynamics. When disengaged ($e = 0$), a teaching action raises engagement with probability $P_{\text{engage}}(a)$ (Table 4). Games have higher engagement probabilities than practice, consistent with game-based learning research (Corbett and Anderson, 1994). When already engaged ($e = 1$),

engagement drops back to 0 with probability 0.5 after each action, reflecting the transient nature of high engagement:

$$p(e' \mid s, a) = \begin{cases} P_{\text{engage}}(a), & e'=1, \quad e=0 \quad \text{(engagement gained)}, \\ 1 - P_{\text{engage}}(a), & e'=0, \quad e=0 \quad \text{(remains disengaged)}, \\ 0.5, & e'=0, \quad e=1 \quad \text{(engagement lost)}, \\ 0.5, & e'=1, \quad e=1 \quad \text{(engagement retained)}. \end{cases} \tag{13}$$

2.6 Transition Diagram

Figures 2 and 3 summarise the transition dynamics. Figure 2 shows the proficiency ladder from Novice (k=0) to Expert (k=9): a teaching action advances one level with success probability ρ_{succ}, earning $+10$ per level, $+20$ on first reaching k=9, and $+5$ for maintaining expertise, while fatigue escalates independently and is cleared by Rest. Figure 3 details the two special dynamics: the engagement toggle, games and practice raise e with probability $P_{\text{engage}}(a)$ and grant a $+0.2$ success bonus while engaged, and the overnight reset, where choosing Rest at h=5 clears fatigue and applies a $50/50$ proficiency-decay event; any non-Rest action during the sleep window incurs the -30 night-violation penalty.

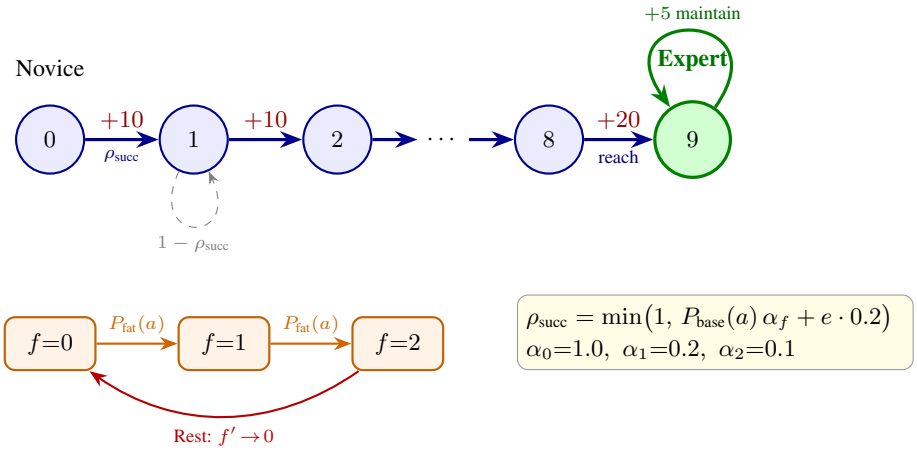

Figure 2: Proficiency and fatigue dynamics. **Top:** the proficiency ladder from Novice (k=0) to Expert (k=9); a teaching action advances one level with success probability ρ_{succ} (dashed self-loop: failure, probability $1 - \rho_{\text{succ}}$), earning $+10$ per level, $+20$ on first reaching k=9, and $+5$ to maintain expertise. **Bottom-left:** fatigue escalates with probability $P_{\text{fat}}(a)$ and is reset to f=0 by Rest. **Bottom-right:** the effective success probability, attenuated by the fatigue scale α_f and boosted by $+0.2$ when engaged.

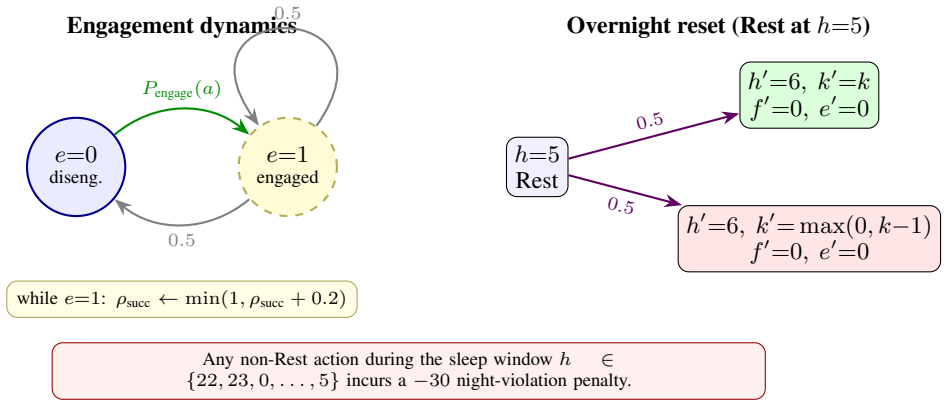

Figure 3: Engagement and overnight dynamics. **Left:** games and practice raise engagement from e=0 to e=1 with probability $P_{\text{engage}}(a)$; engagement decays to 0 with probability 0.5 after any action and grants a +0.2 success bonus while active. **Right:** choosing Rest at h=5 advances to the next morning, clears fatigue, and applies a 50/50 memory-decay event on proficiency.

2.7 Reward Function $\mathcal{R}$

The reward function $\mathcal{R}(s, a, s')$ rewards proficiency milestones and penalises fatigue, sleep-window violations, and the overnight transition:

$$\mathcal{R}(s, a, s') = r_{\text{prog}}(k, k') + r_{\text{fatigue}}(f') + r_{\text{overnight}}(s, a) + r_{\text{night}}(h, a) - c_{\text{step}}, \tag{14}$$

where $c_{\text{step}} = 1$.

Progress component.

$$r_{\text{prog}}(k, k') = \begin{cases} +20, & k < 9 \text{ and } k' = 9 \quad \text{(Reach Expert)}, \\ +10, & k < 9 \text{ and } k' > k \quad \text{(Standard Level Up)}, \\ +5, & k = 9 \text{ and } k' = 9 \quad \text{(Maintained Expert)}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

State and event penalties.

$$r_{\text{fatigue}}(f') = -2 \cdot \mathbb{I}[f' = 1] - 4 \cdot \mathbb{I}[f' = 2], \tag{16}$$

$$r_{\text{overnight}}(s, a) = -7 \cdot \mathbb{I}[h = 5 \wedge a = \text{Rest}], \tag{17}$$

$$r_{\text{night}}(h, a) = -30 \cdot \mathbb{I}[h \in \{22, 23, 0, 1, 2, 3, 4, 5\} \wedge a \neq \text{Rest}]. \tag{18}$$

Table 5: Summary of Reward Function Components.

Event Category	Condition	Value
Reach Expert	$k < 9 \to k' = 9$	+20
Standard Level Up	$k < 9$ and $k' > k$	+10
Maintain Expert	$k = 9 \to k' = 9$	+5
Moderate-Fatigue Penalty	$f' = 1$	−2
High-Fatigue Penalty	$f' = 2$	−4
Overnight Penalty	*Rest* at $h = 5$	−7
Night-Violation Penalty	Non-*Rest* action with h in sleep window	−30
Time Penalty	Applied every step unconditionally	−1

2.8 Assumptions and Design Choices

The MDP encodes four modelling choices that are worth making explicit. (i) **Markovian fatigue.** Fatigue is summarised by a single ordinal variable $f \in \{0, 1, 2\}$ rather than a continuous physiological signal, keeping the state space tractable while capturing graded exhaustion (Sutton and Barto, 2018). (ii) **Soft sleep window.** The sleep window $h \in \{22, 23, 0, 1, 2, 3, 4, 5\}$ is enforced through a large reward penalty (-30) rather than an action mask. The motivation for this design is to model the realistic scenario where a student might choose to study through the night (the so-called "begadang" pattern), and to let the agent *learn* that nocturnal study is counterproductive through reward signals rather than having this behaviour hard-coded. This allows us to study whether deep RL can discover the sleep policy from experience alone, without explicit programming. (iii) **Single overnight reset trigger.** The daily memory-decay event only fires on *Rest* at $h = 5$, forcing the agent to make an explicit decision to end the day. (iv) **Transient engagement.** Engagement is modelled as a binary state that can be raised by stimulating activities but decays stochastically, reflecting the empirical finding that motivational states in game-based learning are transient (Corbett and Anderson, 1994).

2.9 Gymnasium Implementation

The MDP is exposed as a Gymnasium (Towers et al., 2024) environment registered under the identifier `RobotTutor-v2`. The implementation follows the standard `reset`/`step` API and is fully compatible with the Stable-Baselines3 training library. The environment exposes:

- **Observation space.** $\texttt{Box}([0, 0, 0, 0], [23, 9, 2, 1], \texttt{float32})$: the raw state vector (h, k, f, e) cast to floats, representing all four state dimensions.
- **Action space.** $\texttt{Discrete}(9)$: integer indices 0–8 mapping to the nine actions in Table 3.
- **Episode termination.** Episodes are truncated at a configurable step limit (default 240 steps, corresponding to ten full 24-hour days); `terminated` is always `False` as the MDP is continuing.
- **Custom info.** At episode end, the `info` dict contains `ep_first_expert_step` (the step at which $k = 9$ was first reached, or -1 if never) and `ep_fatigue2_count` (the number of steps in which $f = 2$ was observed).

3 Compared Methods

The optimization objective is to find the optimal policy π^* that maximises the expected discounted return from the initial state $s_0 = (h{=}6, k{=}0, f{=}0, e{=}0)$ (Sutton and Barto, 2018):

$$\pi^* \in \arg\max_{\pi} \; \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(S_t, A_t, S_{t+1}) \;\middle|\; S_0 = s_0\right]. \tag{19}$$

We solve this objective with four deep RL methods that learn directly from environment interaction, without requiring access to the analytical transition probability of Section 2. The algorithms cover the canonical axes of modern deep RL: **Deep Q-Network (DQN)** (Mnih et al., 2015) is the value-based, off-policy baseline with experience replay; **Proximal Policy Optimization (PPO)** (Schulman et al., 2017) is the clipped-surrogate on-policy actor-critic method; **Trust Region Policy Optimization (TRPO)** (Schulman et al., 2015) is the constrained policy-gradient method with a monotonic improvement guarantee; and **Soft Actor-Critic (SAC)** (Haarnoja et al., 2018) is the maximum-entropy off-policy actor-critic method. All four algorithms are implemented via the Stable-Baselines3 library and share a two-layer MLP backbone with hidden width 128 and ReLU activations.

3.1 Deep Q-Network (DQN)

DQN (Mnih et al., 2015) replaces the tabular action-value function $q_\gamma(s, a)$ with a neural approximation $q_\gamma(s, a; \theta)$ and adds two stability mechanisms: an experience replay buffer and a slowly-tracking target network with parameters θ^-.

At every environment step, the transition $(S_t, A_t, R_{t+1}, S_{t+1}, \text{done}_t)$ is stored in the replay buffer. A uniformly random mini-batch of size B is sampled to compute the temporal-difference target (Sutton and Barto, 2018):

$$y_i = R_{i+1} + \gamma \cdot \mathbb{I}[\neg\,\text{done}_i] \cdot \max_{a' \in \mathcal{A}} q_\gamma(S_{i+1}, a'; \theta^-). \tag{20}$$

The network is trained to minimise the Huber loss (Smooth L1):

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \text{SmoothL1}\big(q_\gamma(S_i, A_i; \theta) - y_i\big). \tag{21}$$

We use the Huber loss rather than mean-squared error because it provides linear gradients for large prediction errors, making training more robust to the occasional large-magnitude penalties (e.g., the -30 night-violation reward) that can appear in the replay buffer early in training, while behaving like MSE for small errors and thereby preserving fast, stable convergence near the optimum (Mnih et al., 2015).

After each gradient step, the target parameters are softly updated by Polyak averaging:

$$\theta^- \leftarrow \tau\,\theta + (1 - \tau)\,\theta^-. \tag{22}$$

Rather than periodically copying the online parameters to the target network in a hard update, Polyak averaging (exponential moving average with coefficient τ) smoothly tracks the evolving online network, preventing the target from jumping abruptly and thereby reducing the oscillatory instability that hard target updates can introduce when the state distribution is rapidly changing. We set $\tau = 0.005$, so the target network absorbs only 0.5% of each online update; this provides a stable regression target throughout training while still tracking policy improvements on a timescale of thousands of steps.

3.2 Proximal Policy Optimization (PPO)

PPO (Schulman et al., 2017) is an on-policy actor-critic algorithm that improves training stability by replacing the hard KL constraint of TRPO with a clipped surrogate objective. At each update, PPO collects a rollout of n_{steps} environment steps under the current policy $\pi_{\theta_{\text{old}}}$, computes Generalised Advantage Estimation (GAE) (Schulman et al., 2016) advantages A_t, and then optimises the clipped objective over multiple epochs:

$$\mathcal{L}_{\text{CLIP}}(\theta) = \mathbb{E}_t\big[\min(r_t(\theta)\,A_t,\ \text{clip}(r_t(\theta),\,1-\varepsilon,\,1+\varepsilon)\,A_t)\big]\,, \tag{23}$$

where $r_t(\theta) = \pi_\theta(A_t \mid S_t)/\pi_{\theta_{\text{old}}}(A_t \mid S_t)$ is the probability ratio and ε is the clip range. The clipping prevents excessively large policy updates without requiring the expensive conjugate-gradient solve of TRPO, making PPO significantly faster per update while still bounding the step size in probability space. A separate value-function head $\hat{V}_\gamma(s;\phi)$ is trained jointly by MSE regression toward the discounted return, and an entropy bonus $\beta_H \mathcal{H}[\pi_\theta]$ encourages exploration throughout training.

3.3 Trust Region Policy Optimization (TRPO)

TRPO (Schulman et al., 2015) is a constrained policy-gradient method with a categorical policy $\pi_\theta(a \mid s)$ and a separate state-value baseline $\hat{V}_\gamma(s;\phi)$. Advantages are estimated via GAE (Schulman et al., 2016):

$$\delta_t = \mathcal{R}(S_t, A_t, S_{t+1}) + \gamma\,\hat{V}_\gamma(S_{t+1};\phi) - \hat{V}_\gamma(S_t;\phi), \qquad A_t = \sum_{k=0}^{T-t-1} (\gamma\lambda)^k\,\delta_{t+k}. \tag{24}$$

The policy update maximises the importance-sampled surrogate subject to a KL trust-region constraint:

$$\max_{\theta} \; \mathcal{L}_{\text{surr}}(\theta) = \mathbb{E}\left[\frac{\pi_\theta(A_t \mid S_t)}{\pi_{\theta_{\text{old}}}(A_t \mid S_t)} A_t\right] \quad \text{s.t.} \quad \mathbb{E}\big[\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot \mid S_t) \,\big\|\, \pi_\theta(\cdot \mid S_t)\big)\big] \le \delta. \tag{25}$$

Each outer step is solved approximately with conjugate gradient on the Fisher vector product, followed by a backtracking line search that enforces both the KL constraint and a minimum surrogate improvement ratio. Compared to DQN, the explicit KL constraint removes the need for a target network or replay buffer. Compared to PPO, the trust region provides a monotonic improvement guarantee on the surrogate $\mathcal{L}_{\text{surr}}(\theta)$, which lower-bounds the true improvement in expected return when the KL constraint is satisfied (Schulman et al., 2015).

3.4 Soft Actor-Critic (SAC)

SAC (Haarnoja et al., 2018) is an off-policy actor-critic algorithm that augments the standard return with a policy-entropy term, optimising the maximum-entropy objective

$$J(\pi_\theta) = \sum_{t=0}^{\infty} \mathbb{E}_{(S_t, A_t) \sim \rho_{\pi_\theta}} \Big[\gamma^t \Big(\mathcal{R}(S_t, A_t, S_{t+1}) + \alpha\, \mathcal{H}[\pi_\theta(\cdot \mid S_t)]\Big)\Big], \tag{26}$$

where $\mathcal{H}[\pi_\theta(\cdot \mid S_t)] = -\,\mathbb{E}_{a \sim \pi_\theta}[\log \pi_\theta(a \mid S_t)]$ is the entropy of the policy at state S_t. The temperature α trades off reward against exploration and is tuned automatically, so that the agent acts as randomly as possible while still solving the task. Like DQN, SAC learns off-policy from a replay buffer; like the policy-gradient methods, it maintains an explicit stochastic policy. It trains twin soft Q-functions (clipped double-Q to curb overestimation), a policy network optimised by the reparameterised policy gradient, and soft target networks updated by Polyak averaging with coefficient τ.

Discrete-to-continuous action interface. The reference Stable-Baselines3 implementation of SAC is defined only for continuous (`Box`) action spaces, whereas the Robot Tutor MDP uses the discrete nine-action set of Table 3. To keep SAC unmodified while comparing it on the identical task, we wrap the environment so that the agent emits a continuous vector $u \in [-1, 1]^9$ — one logit per action — and the executed discrete action is selected by $a = \arg\max_i u_i$. The wrapper alters only the action interface; the underlying transition dynamics, reward function, and `info` diagnostics are unchanged, so all four algorithms are trained and evaluated on the same MDP. This argmax embedding is the simplest faithful way to expose a discrete task to a continuous-only learner; we note it can dampen SAC's entropy signal, since many distinct continuous vectors collapse to the same discrete action.

4 Experimental Setup

Hardware and software. All experiments were executed locally on an AMD Ryzen 7 8845HS with 16 GB RAM, using Python 3.13, PyTorch (Paszke et al., 2019), and the Stable-Baselines3 library (v2.8.0). Every algorithm is accessed through Stable-Baselines3's unified API on top of the `RobotTutor-v2` Gymnasium environment described in Section 2.9.

Training protocol. Each algorithm is trained for 3,000 episodes (equivalent to 720,000 environment steps at 240 steps per episode) under four independent random seeds $\{0, 1, 2, 3\}$. All results are reported as mean $\pm$ standard deviation across seeds. Every episode is initialised from $s_0 = (h{=}6, k{=}0, f{=}0, e{=}0)$ with discount factor $\gamma = 0.95$. SAC, which is defined for continuous action spaces, is trained on the same MDP through a continuous-action wrapper that maps a nine-dimensional logit vector to a discrete action by argmax (Section 3.4).

Evaluation metrics. Beyond mean episode reward, we track three additional metrics per episode:

- **Steps to first expert proficiency.** The environment step at which $k = 9$ is reached for the first time within an episode (-1 if never reached).
- **High-fatigue exposure.** The number of steps per episode in which $f = 2$, measuring how often the agent pushes the student into the highest fatigue level.

- **DQN training loss.** The per-gradient-step Huber loss, reported smoothed with a 200-step moving average (Appendix D).

Hyperparameters. All hyperparameters are centralised in a YAML configuration file. Table 6 consolidates the algorithm-specific settings.

Table 6: Hyperparameter configuration for the four deep RL algorithms. SAC uses a single learning rate for its actor and twin critics, an automatically tuned entropy temperature, and trains on the discrete MDP through the continuous-action wrapper of Section 3.4.

Hyperparameter	**DQN**	**PPO**	**TRPO**	**SAC**
Discount factor γ	0.95	0.95	0.95	0.95
Hidden width	128	128	128	128
Network layers	2	2	2	2
Training episodes	3000	3000	3000	3000
Seeds	4	4	4	4
Q-network learning rate	10^{-4}	-	-	-
Policy learning rate	-	3×10^{-4}	10^{-3}	3×10^{-4}
Replay buffer capacity	10^5	-	-	10^5
Learning starts	5000	-	-	5000
Mini-batch size B	64	64	128	64
Target-network rate τ	0.005	-	-	0.005
ε-greedy schedule	$1.0 \rightarrow 0.05$ over 15% of steps	-	-	-
Rollout steps n_{steps}	-	2048	2048	-
Optimisation epochs per rollout	-	10	-	-
Clip range ε	-	0.2	-	-
Entropy coefficient β_H	-	0.01	-	auto
GAE λ	-	0.95	0.95	-
KL trust region budget δ	-	-	0.01	-
Conjugate gradient iterations	-	-	10	-
Fisher vector damping	-	-	0.1	-
Critic update epochs	-	-	10	-
Gradient steps / train freq.	-	-	-	1/4

The 10-day bootcamp analogy. Because each time step corresponds to one hour of real time, a 240-step episode maps to a continuous *ten-day intensive bootcamp* ($10 \times 24 = 240$ hours). Every episode begins on the morning of Day 1 at h=6 with a fresh, novice, disengaged student (k=0, f=0, e=0), and the agent must orchestrate ten days of teaching, practice, games, and rest to maximise the student's proficiency by the end of Day 10.

5 Results and Analysis

5.1 Learning Curves

Figure 4 shows the training reward curves; Figures 7–9 summarise the three final-performance metrics as bar charts, while Figures 5 and 6 track steps-to-expert and high-fatigue exposure over training. The DQN training loss is reported in Appendix D. Table 7 then reports a *held-out evaluation* in which every policy is run for 200 fresh episodes per seed: alongside the four learned algorithms we evaluate two non-learned baselines, a **Random** policy and a fixed **Deterministic** schedule, under the identical protocol, and additionally record how the agent's 240 steps per episode are distributed over the four action segments (Lecture, Game, Practice, Rest). These baselines are never trained; they participate only in the final evaluation as reference points against which the learned policies are compared.

Table 7: Held-out evaluation on the `RobotTutor-v2` MDP. Each policy is run for 200 fresh episodes per seed across the four seeds; every cell shows the mean with the cross-seed standard deviation as a small subscript. The four learned policies act stochastically, exactly as during the final 200 training episodes; **Random** (uniform action every step) and **Deterministic** (a fixed hand-crafted schedule, rest at night, and a repeating daytime cycle of lectures, practice, and games) are non-learned baselines evaluated under the same protocol. The metrics are defined as follows: *Steps to Expert* is the step at which proficiency first reaches the expert level $k = 9$, averaged only over episodes that reach it; *Expert Rate* is the fraction of episodes that *ever* reach $k = 9$ within the 240-step horizon; and *Fat-2 / Ep* is the mean number of steps per episode the student spends at the maximum fatigue level $f = 2$. The right-hand block reports the mean steps per episode spent on each action segment, Lecture $= \{$theoretical, practical$\}$, Game $= \{$hard, medium, easy$\}$, Practice $= \{$hard, medium, easy$\}$, Rest, which sum to the 240-step horizon. Crucially, although *every* policy (even Random) eventually reaches $k = 9$ in all episodes, a 240-step horizon leaves ample room for chance success, so the Expert Rate column is uninformative on its own, the learned policies do so roughly twice as fast (lower Steps to Expert) and at a tiny fraction of the high-fatigue exposure: SAC, DQN, and TRPO each keep Fat-2 / Ep below 4.4, whereas the Deterministic schedule reaches 37.3 and Random a punishing 144.7. SAC attains the best reward overall; both baselines trail every learned policy by a wide margin, confirming that the learned policies are non-trivial.

	Performance				**Action segments (steps / episode)**			
Policy	**Reward**	**Steps to Expert**	**Expert Rate**	**Fat-2 / Ep**	**Lecture**	**Game**	**Practice**	**Rest**
DQN	$757.7_{\pm 11.6}$	$24.4_{\pm 0.7}$	100%	$0.41_{\pm 0.18}$	$25.2_{\pm 8.4}$	$4.1_{\pm 0.2}$	$5.0_{\pm 1.0}$	$205.7_{\pm 8.5}$
PPO	$309.1_{\pm 436.8}$	$72.0_{\pm 38.2}$	68.9%	$0.00_{\pm 0.00}$	$18.4_{\pm 2.6}$	$0.0_{\pm 0.0}$	$1.8_{\pm 2.9}$	$219.8_{\pm 3.1}$
TRPO	$740.7_{\pm 31.9}$	$19.9_{\pm 0.3}$	100%	$4.38_{\pm 7.26}$	$107.8_{\pm 20.6}$	$0.1_{\pm 0.0}$	$0.0_{\pm 0.0}$	$132.1_{\pm 20.5}$
SAC	$\mathbf{826.8}_{\pm 17.0}$	$22.1_{\pm 1.6}$	100%	$0.68_{\pm 0.63}$	$57.9_{\pm 12.1}$	$9.6_{\pm 13.5}$	$1.8_{\pm 1.2}$	$170.6_{\pm 21.9}$
Random	$-1966.5_{\pm 5.1}$	$44.1_{\pm 0.3}$	100%	$144.71_{\pm 0.29}$	$53.7_{\pm 0.5}$	$80.0_{\pm 0.5}$	$79.6_{\pm 0.4}$	$26.7_{\pm 0.2}$
Deterministic	$507.7_{\pm 2.3}$	$49.0_{\pm 0.7}$	100%	$37.33_{\pm 0.25}$	$30.0_{\pm 0.0}$	$43.0_{\pm 0.0}$	$43.0_{\pm 0.0}$	$124.0_{\pm 0.0}$

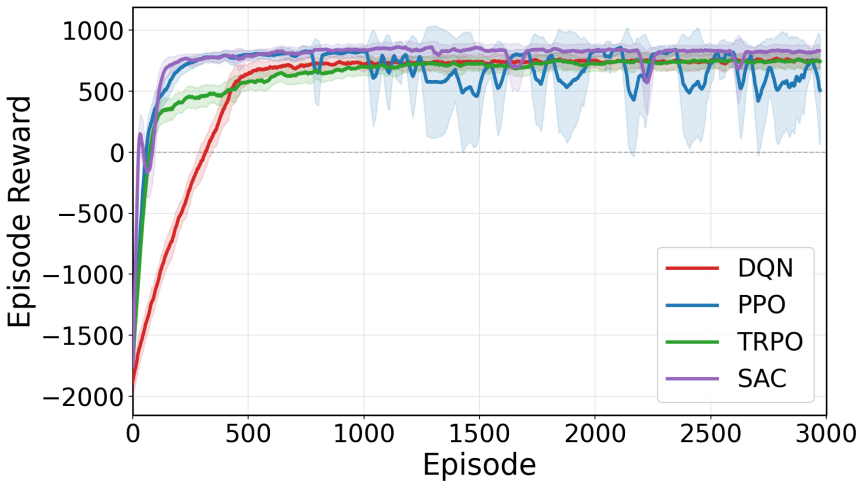

Figure 4: Training reward per episode (mean ± shaded std across 4 seeds, 30-episode moving average). SAC reaches the highest reward; PPO is the most variable across seeds.

5.2 Analysis

Most algorithms solve the task reliably, but PPO does not. Three of the four algorithms, DQN, TRPO, and SAC, reach expert proficiency ($k = 9$) in 100% of their held-out evaluation episodes (Table 7), confirming that the 1440-state `RobotTutor-v2` MDP remains tractable for neural function approximation even with a ten-level proficiency ladder and a 240-step horizon. PPO is the exception: it reaches $k = 9$ in only 68.9% of held-out episodes and needs the most steps to do so (72.0 ± 38.2), indicating that it does not reliably drive the student all the way to mastery. For the three reliable algorithms expert is reached in roughly 20–24 steps, even though the ladder now requires nine successful level-ups rather than four.

SAC is the best-performing algorithm overall. SAC attains the highest held-out reward (826.8 ± 17.0), reaches expert in 100% of episodes, and converges quickly during training (≈ 236 episodes to 90% of its final reward; Figure 4). Its maximum-entropy objective and off-policy replay let it explore the long 240-step horizon efficiently while keeping high-fatigue exposure low (0.68 steps per episode). Notably, SAC outperforms all three natively discrete algorithms *despite* acting through the argmax wrapper, which discards the magnitude of its continuous outputs, evidence that an off-policy maximum-entropy learner is well suited to this long-horizon sequencing problem.

DQN and TRPO are strong and stable. DQN (757.7 ± 11.6) and TRPO (740.7 ± 31.9) follow closely, both reaching expert in every episode; DQN in fact has the *lowest* cross-seed variance of all methods. TRPO is the fastest to first reach expert (19.9 ± 0.3 steps), while DQN converges in fewer training episodes (≈ 589 vs. ≈ 932), reflecting the sample efficiency of off-policy replay against TRPO's conservative trust-region updates. The DQN training loss (Figure 14, Appendix) shows the characteristic early spike as large-magnitude night-violation penalties (-30) enter the replay buffer, followed by a monotonic decay as the target network stabilises.

PPO is the least reliable on the harder task. In contrast to the reduced five-level task, where on-policy PPO was competitive, PPO is now the weakest learned method: its held-out reward is both the lowest (309.1) and by far the most variable (± 436.8), and it is the only learned algorithm that fails to reach expert in every episode (68.9%). Its held-out reward even falls below its average over the final training episodes, exposing how brittle the policy is across seeds: the longer horizon and deeper proficiency ladder make the on-policy gradient signal noisier and the policy more prone to collapse on some seeds. Its apparently fast convergence during training (≈ 190 episodes; Figure 4) is therefore misleading, since the plateau it reaches early is itself low and high-variance. PPO does achieve the lowest high-fatigue exposure (0.00 steps per episode), but this partly reflects *under-teaching*: by acting cautiously it both avoids fatigue and sometimes stops short of mastery.

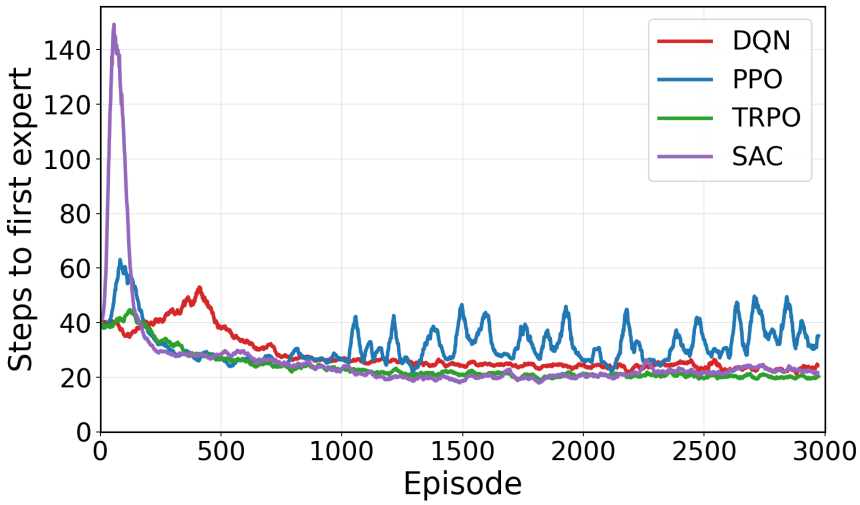

Figure 5: Steps to first expert proficiency over training (rolling mean, 30-episode window; episodes that never reach $k = 9$ are excluded).

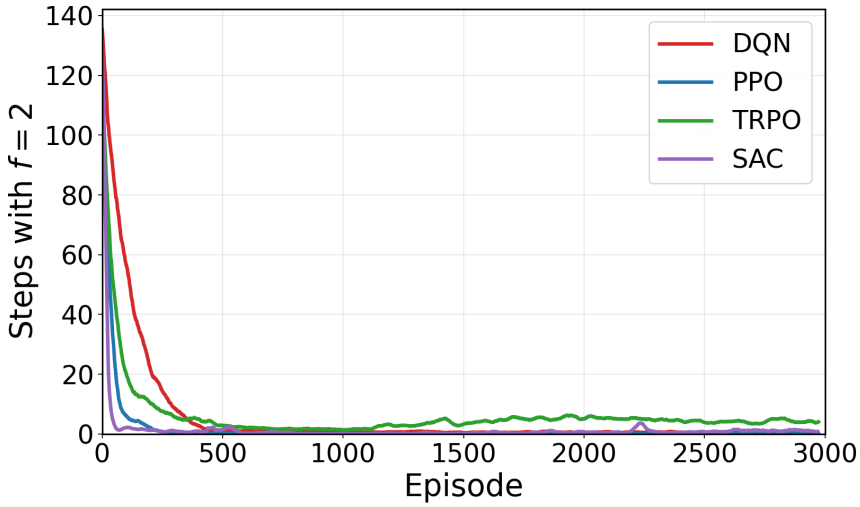

Figure 6: High-fatigue ($f = 2$) exposure per episode over training (rolling mean, 30-episode window).

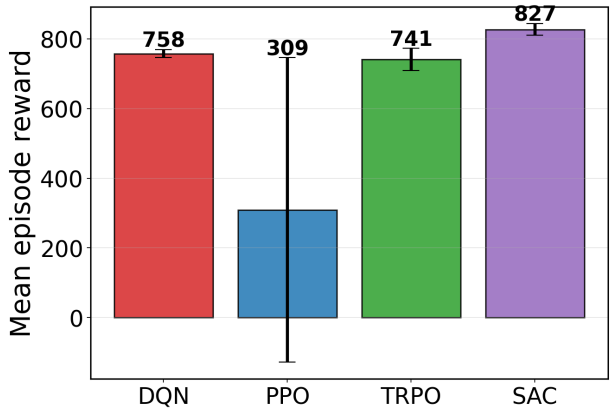

Figure 7: Mean episode reward in the held-out evaluation (200 episodes per seed, mean $\pm$ std across 4 seeds).

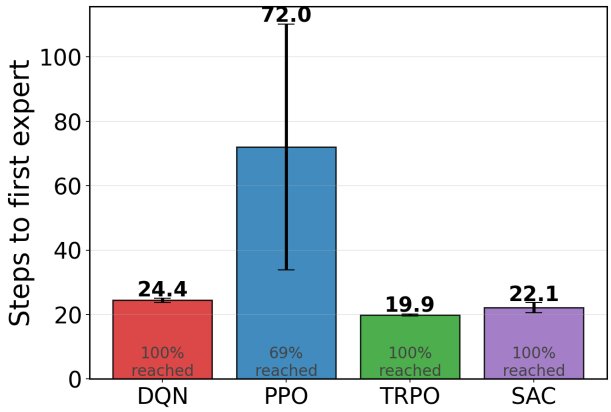

Figure 8: Mean steps to first expert proficiency in the held-out evaluation (200 episodes per seed), with the percentage of episodes in which $k = 9$ is reached.

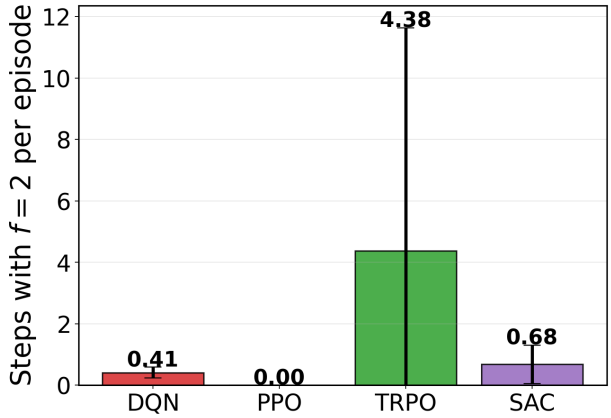

Figure 9: Mean high-fatigue ($f = 2$) steps per episode in the held-out evaluation (200 episodes per seed).

Fatigue management exposes a proficiency–fatigue trade-off. High-fatigue ($f = 2$) exposure differs sharply across the learned methods (Table 7): PPO (0.00) and DQN (0.41) keep the student out of the highest fatigue level almost entirely, SAC is low (0.68), and TRPO is the highest and most variable (4.38 ± 7.26). TRPO's aggressive proficiency gains, it reaches expert in the fewest steps, come at the cost of pushing the student through high-fatigue states more often, illustrating the proficiency–fatigue trade-off at the heart of the MDP.

Learned policies dominate the non-learned baselines. The two reference baselines confirm that the task is far from trivial. The **Random** policy is catastrophic (-1966.5): acting indiscriminately, including teaching during night hours, it incurs constant night-violation penalties and drives the student into high fatigue on 144.7 of every 240 steps. The **Deterministic** schedule fares much better (507.7) by resting at night and cycling through a balanced activity sequence, yet it still trails every learned policy by a wide margin and accumulates 37.3 high-fatigue steps per episode. Tellingly, *both* baselines do eventually reach $k = 9$ in every episode, a 240-step horizon leaves ample room for chance success, but they take roughly twice as many steps to first reach expert (44–49 vs. 20–24) and earn a small fraction of the reward. Reaching mastery is therefore easy; reaching it *efficiently and safely* is what the learned policies provide.

How the agents spend their time. The action-segment columns of Table 7 reveal markedly different teaching styles. All learned policies spend most of the horizon resting (DQN 205.7, SAC 170.6, TRPO 132.1 steps per episode): once the student is expert, further teaching risks fatigue penalties for only a small maintenance reward, so the agents protect the student by resting. Among

teaching segments, lectures dominate, they carry the highest base learning rate at the lowest fatigue cost, with TRPO relying on them most heavily (107.8 steps) and consequently incurring the most fatigue, while games and practice are used sparingly. By contrast the Random and Deterministic baselines spread their steps far more evenly across Game and Practice (≈ 80 and ≈ 43 steps each, respectively), which is precisely what drives their elevated fatigue and depressed reward.

Recommended algorithm. For the Robot Tutor problem, **SAC is the recommended algorithm**: it achieves the highest held-out reward, reaches expert in every episode, converges quickly, and keeps fatigue low. DQN is the best natively discrete choice, it is the most stable across seeds and among the most fatigue-sparing of the reliable methods; TRPO is preferable when reaching mastery in the fewest steps matters more than fatigue exposure. PPO is not recommended for this longer-horizon, ten-level task: it is unstable across seeds and does not reliably reach expert proficiency. All four learned policies, however, comfortably dominate the Random and Deterministic baselines.

6 Conclusion

6.1 Key Findings

This paper formalised the Robot Tutor problem as a 1440-state MDP over a 24-hour clock, a ten-level proficiency variable, a three-level fatigue variable, and a binary engagement variable, and solved it with four deep RL algorithms (DQN, PPO, TRPO, and SAC) operating on a single shared `RobotTutor-v2` Gymnasium interface across four independent random seeds.

The empirical comparison yields four principal takeaways.

First, **three of the four algorithms reliably solve the tutoring task**: DQN, TRPO, and SAC each achieve a 100% expert-proficiency reach rate in the held-out evaluation episodes, demonstrating that neural function approximation scales to the 1440-state MDP with a ten-level proficiency ladder; PPO is the lone exception, reaching expert in only 68.9% of episodes.

Second, **SAC is the best-performing algorithm overall**. It achieves the highest held-out reward (826.8 ± 17.0), a 100% expert-reach rate, and fast convergence (≈ 236 episodes), all while keeping high-fatigue exposure low. That an off-policy maximum-entropy learner, trained through a continuous-to-discrete argmax wrapper, beats the natively discrete methods is the clearest single result of the study.

Third, **the off-policy and trust-region methods are stable, whereas PPO is not**. DQN (757.7 ± 11.6, the lowest cross-seed variance of all methods) and TRPO (740.7 ± 31.9) trail SAC closely, with full expert-reach, while PPO collapses to the lowest and most variable reward (309.1 ± 436.8). The deeper proficiency ladder and longer 240-step horizon make PPO's on-policy gradient signal noisier and its policy prone to seed-dependent collapse, reversing the ordering observed on the easier five-level task.

Fourth, **fatigue management reveals a proficiency–fatigue trade-off**. PPO and DQN keep high-fatigue exposure near zero (0.00 and 0.41 steps per episode), SAC is low (0.68), and TRPO is the highest (4.38): TRPO reaches mastery in the fewest steps but pushes the student through high-fatigue states most often. PPO's low fatigue partly reflects under-teaching rather than skilful pacing, since it does not always reach mastery. Finally, all four learned policies decisively outperform the Random and Deterministic baselines, which reach mastery far more slowly and at much lower reward.

6.2 Future Works

Three directions follow naturally from this study. First, replace the discrete proficiency variable with a continuous mastery score derived from item response theory (Corbett and Anderson, 1994); this is the regime where function approximation becomes strictly necessary and where the engagement dynamics modelled here become even more impactful. Second, extend the formulation to a Partially Observable MDP (POMDP) in which fatigue and engagement are latent states inferred from observable proxies (response latency, error rates), which can be addressed with recurrent policy networks on top of the PPO or TRPO training loop. Third, train a single agent across a population of students with heterogeneous learning rates and fatigue thresholds, drawing on the adaptive personalisation

methodology of Ruan et al. (2024) and Chung et al. (2026); the cross-algorithm ranking established here (SAC $>$ DQN $\approx$ TRPO $>$ PPO) serves as the calibrated baseline for any such extension.

References

Bassen, J., Balaji, B., Schaarschmidt, M., Thille, C., Painter, J., Zimmaro, D., Games, A., et al. (2020). Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Chung, A. T.-H., Zhang, B., Kung, L.-C., Bastani, H., and Bastani, O. (2026). Effective personalized AI tutors via LLM-guided reinforcement learning. SSRN Working Paper, Stanford SCALE Lab.

Corbett, A. T. and Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870.

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Ruan, S., Nie, A., Steenbergen, W., Jiang, L., Gerber, E., Piech, C., Koenig, S., Kim, J., Schwartz, D., and Haber, N. (2024). Reinforcement learning tutor better supported lower performers in a math task. *Machine Learning*, 113:3401–3421.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Towers, M. et al. (2024). Gymnasium: A standard interface for reinforcement learning environments. `https://gymnasium.farama.org/`.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Appendix

A REINFORCE with Baseline

REINFORCE with Baseline (Williams, 1992; Sutton and Barto, 2018) is an on-policy policy-gradient method that directly optimises the discounted-return objective, parameterised by a categorical policy $\pi_\theta(a \mid s)$. After each episode, the agent computes the empirical discounted return

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} \mathcal{R}(S_k, A_k, S_{k+1}), \tag{27}$$

and uses a learned state-value baseline $\hat{V}_\gamma(s; \phi)$ to reduce variance (Sutton and Barto, 2018). The advantage at step t is

$$A_t = G_t - \hat{V}_\gamma(S_t; \phi). \tag{28}$$

The policy is updated with the REINFORCE estimator augmented by an entropy regulariser:

$$\nabla_\theta V_\gamma(\pi_\theta, s_0) \approx \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t \mid S_t)\, A_t\right] + \beta_H \nabla_\theta \mathcal{H}[\pi_\theta]. \tag{29}$$

The value network is trained simultaneously by mean-squared-error regression:

$$\mathcal{L}_V(\phi) = \frac{1}{T} \sum_{t=0}^{T-1} \big(\hat{V}_\gamma(S_t; \phi) - G_t\big)^2. \tag{30}$$

REINFORCE is included here as an Appendix method because it is the simplest policy-gradient baseline; the main experiments focus on DQN, PPO, TRPO, and SAC as the four primary compared methods.

B Reduced MDP, Classical Baselines, and the Medium Environment

This appendix collects the auxiliary material that supports (but is not the primary focus of) the main study. We document (i) the reduced 12-state MDP used for exhaustive policy enumeration, (ii) the classical Dynamic Programming and tabular RL baselines (Value Iteration, Policy Iteration, Q-Learning), (iii) a smaller 90-state Gymnasium variant of the Robot Tutor environment, and (iv) the experimental results from the classical methods on both environments.

B.1 Reduced MDP Formulation (12 States)

To make exhaustive policy search computationally tractable while preserving the core pedagogical dynamics, we apply aggressive dimensionality reduction: time of day into binary phases $h \in \{0, 1\}$, proficiency into 3 tiers $k \in \{0, 1, 2\}$, and fatigue at $f \in \{0, 1\}$, yielding

$$|\mathcal{S}_{\text{reduced}}| = 2 \times 3 \times 2 = 12 \text{ states}. \tag{31}$$

With $|\mathcal{A}| = 4$, the total number of deterministic policies is

$$|\Pi| = 4^{12} = 16{,}777{,}216, \tag{32}$$

which is large but enumerable. Exhaustive policy search is the only method evaluated on this MDP and serves as a brute-force ground-truth check.

Table 8: Comparison between the two classical Robot Tutor Gymnasium environments.

Aspect	RobotTutor-med	RobotTutor-large
Hour h	$\{0, \ldots, 8\}$ (9 hours)	$\{0, \ldots, 23\}$ (24 hours)
Fatigue f	2 levels (fresh / tired)	3 levels (fresh / moderate / high)
Overnight transition	$h = 8 \to h = 0$, deterministic	$h = 5 \to h = 6$ only when Rest is taken
Sleep window	–	$h \in \{22, 23, 0, \ldots, 5\}$, -30 if not Rest
Reach-expert reward	$+30$	$+20$
Maintain-expert reward	$+15$	$+5$
$f' = 1$ penalty	-4	-2
$f' = 2$ penalty	–	-4
Overnight penalty	-10	-7
Episode horizon	45 steps	120 steps
$\lvert\mathcal{S}\rvert$	90	360

B.2 Exhaustive Policy Search

Let $P_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ be the state-transition matrix induced by π, and $\vec{r}_\pi$ the expected immediate-reward vector. The state-value vector satisfies (Sutton and Barto, 2018; Puterman, 1994):

$$\vec{V}_\gamma(\pi) = (I - \gamma P_\pi)^{-1}\vec{r}_\pi. \tag{33}$$

Exhaustive search materialises every deterministic policy, evaluates via the above closed form, and records the maximiser.

B.3 Dynamic Programming and Tabular Q-Learning

Value Iteration (VI). Value Iteration (Puterman, 1994) applies the Bellman optimality operator:

$$V_{k+1}(s) = \max_{a\in\mathcal{A}} \left[\mathbb{E}[r(s,a)] + \gamma \sum_{s'} p(s' \mid s,a)\, V_k(s')\right], \tag{34}$$

iterated until $\max_s |V_{k+1}(s) - V_k(s)| < 10^{-4}$.

Policy Iteration (PI). Policy Iteration (Puterman, 1994) alternates between evaluating π_k via the Bellman expectation equation and improving it greedily:

$$\pi_{k+1}(s) = \arg\max_{a\in\mathcal{A}} \left[\mathbb{E}[r(s,a)] + \gamma \sum_{s'} p(s' \mid s,a)\, V^{\pi_k}(s')\right]. \tag{35}$$

Tabular Q-Learning. Q-Learning (Sutton and Barto, 2018) updates the action-value estimate via:

$$q^{k+1}(s,a) \leftarrow q^k(s,a) + \alpha\left(r + \gamma \max_{a'\in\mathcal{A}} q^k(s',a') - q^k(s,a)\right). \tag{36}$$

B.4 Medium Variant of the Gymnasium Environment

A smaller `RobotTutor-med` variant ($|\mathcal{S}| = 90$) is implemented for analytic sanity checks: $h \in \{0, \ldots, 8\}$, $k \in \{0, \ldots, 4\}$, $f \in \{0, 1\}$, with a 45-step episode horizon and coarser reward values.

B.5 Reduced MDP Results: Exhaustive Policy Search

The exhaustive search algorithm evaluated all 16,777,216 candidate policies on the 12-state MDP. The strictly optimal policy π^* yielded

Table 9: Optimal policy mapping $\pi^*(s)$ for the reduced Robot Tutor MDP (12 states).

Phase (h)	**State (h, k, f)**	**Optimal Action**
Day ($h = 0$)	(0, 0, 0)	Lecture
	(0, 0, 1)	Rest
	(0, 1, 0)	Lecture
	(0, 1, 1)	Lecture
	(0, 2, 0)	Rest
	(0, 2, 1)	Rest
Night ($h = 1$)	(1, 0, 0)	Lecture
	(1, 0, 1)	Lecture
	(1, 1, 0)	Lecture
	(1, 1, 1)	Lecture
	(1, 2, 0)	Lecture
	(1, 2, 1)	Lecture

Table 10: Robot Tutor policy evaluation on the analytical 90-state MDP (mean $\pm$ std).

Method	**Discounted**		**Average**	
	Avg Reward	**Disc Reward**	**Avg Reward**	**Disc Reward**
Value Iteration	9.98 ± 1.39	147.84 ± 38.38	9.98 ± 1.39	147.84 ± 38.38
Policy Iteration	9.75 ± 1.50	140.78 ± 40.45	9.75 ± 1.50	140.78 ± 40.45
Q-Learning	9.85 ± 1.50	143.86 ± 38.02	9.72 ± 1.65	140.49 ± 41.26

Table 11: Algorithm runtime comparison on the 90-state MDP (seconds).

Method	**Discounted**	**Average**
Value Iteration	0.317	0.689
Policy Iteration	0.064	0.066
Q-Learning	16.665	18.666

Table 12: Pairwise policy agreement (%) on the 90-state MDP.

Pair	**Discounted**	**Average**
VI vs PI	100.00	100.00
VI vs QL	86.67	83.33
PI vs QL	86.67	83.33

Self-Agreement	**Disc vs Avg**
Value Iteration (VI)	100.00
Policy Iteration (PI)	100.00
Q-Learning (QL)	82.22

$$V^*(\pi^*) = 46.3258. \tag{37}$$

B.6 Analytical 90-State MDP: DP and Tabular Q-Learning

All runs use $\gamma = 0.95$ and random seed 2004.

VI and PI converge to a globally optimal policy under both criteria (mutual agreement 100%). Q-Learning approximates the same policy with $\sim 86.67\%$ overlap in the discounted setting; disagreements are concentrated on boundary states where action values are nearly tied.

Table 13: Gym classical-RL results on `RobotTutor-med` ($|\mathcal{S}| = 90$).

Algorithm	Mean Reward	Std	Runtime (s)	Episodes / Iter.
Q-Learning	371.89	72.59	1.75	1500 episodes
Value Iteration	433.14	72.87	0.22	208 iterations
Policy Iteration	433.14	72.87	0.09	3 iterations

Table 14: Gym classical-RL results on `RobotTutor-large` ($|\mathcal{S}| = 360$). VI/PI value $\bar{R}^* = 975.70$ is the planning ceiling.

Algorithm	Mean Reward	Std	Runtime (s)	Episodes / Iter.
Q-Learning	933.62	60.18	9.18	3000 episodes
Value Iteration	975.70	34.34	0.91	228 iterations
Policy Iteration	975.70	34.34	0.33	5 iterations

B.7 Gym Classical-RL Results

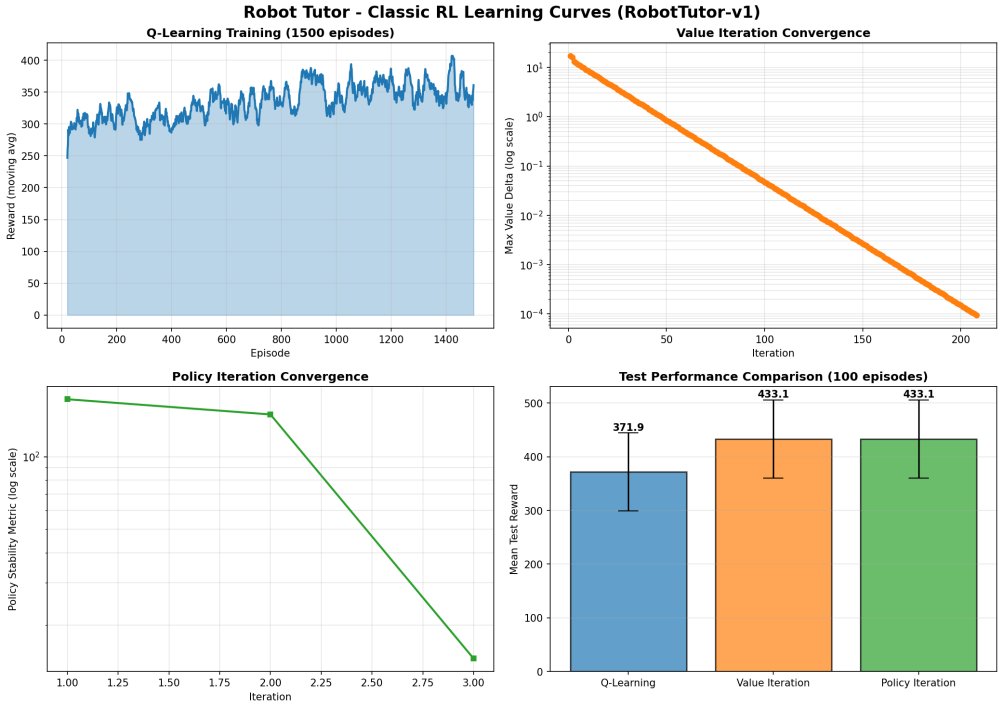

Figure 10: Classical-RL diagnostic curves on `RobotTutor-med`: (top-left) Q-Learning training reward; (top-right) Value Iteration max delta, log scale; (bottom-left) Policy Iteration stability metric; (bottom-right) test-reward bar comparison.

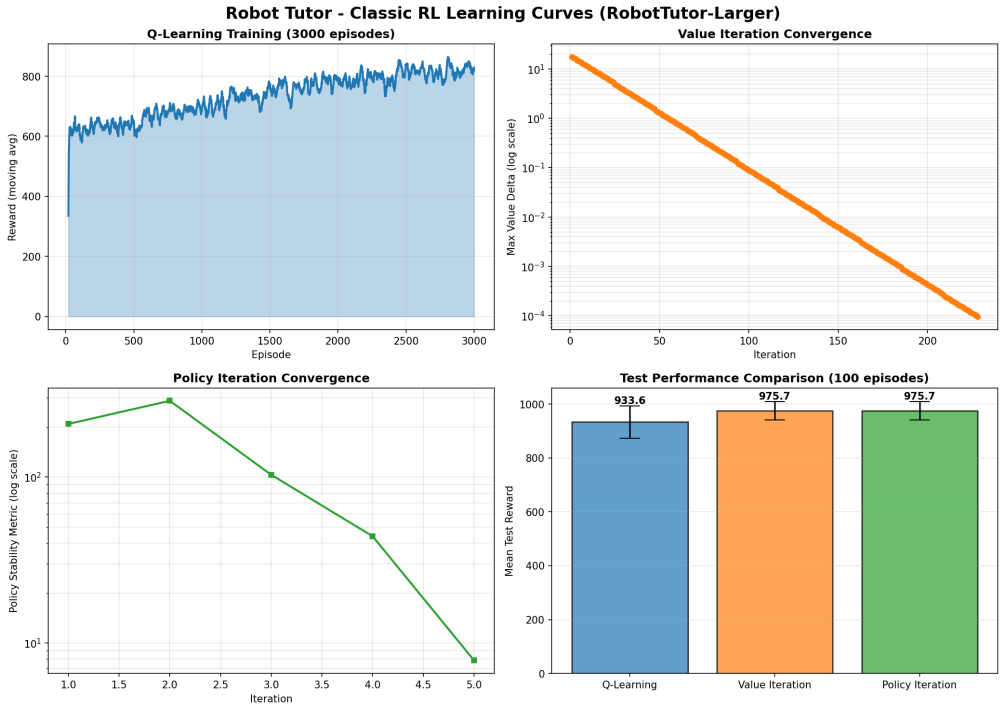

Figure 11: Classical-RL diagnostic curves on `RobotTutor-large`: same panel layout as Figure 10.

B.8 Per-Algorithm Deep RL Dashboards

Figures 12 and 13 show per-algorithm training dashboards for DQN and TRPO on the `RobotTutor-large` environment (pre-engagement-state version, included for historical reference).

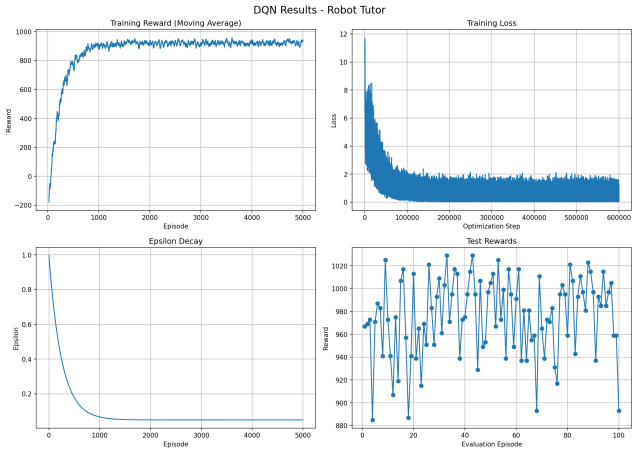

Figure 12: DQN training dashboard on the Robot Tutor MDP (pre-engagement variant).

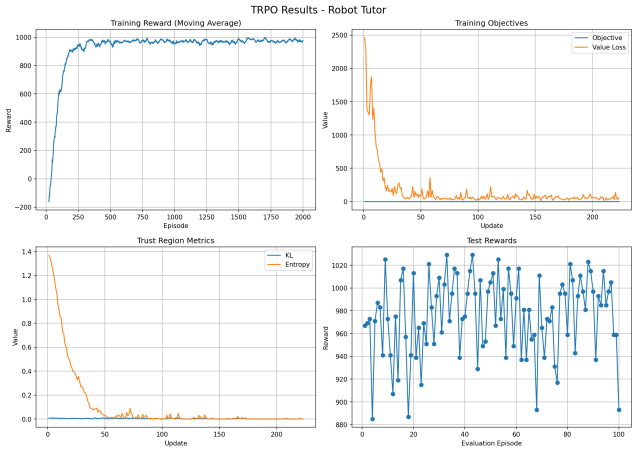

Figure 13: TRPO training dashboard on the Robot Tutor MDP (pre-engagement variant).

C Additional Reference: Bayesian Knowledge Tracing

Corbett and Anderson (1994) introduce Bayesian Knowledge Tracing (BKT), the foundational probabilistic model for estimating student knowledge state from observable correctness sequences. BKT defines four parameters per skill: initial knowledge probability, learning rate, slip, and guess; large-scale empirical studies report learning rates in the range 0.35–0.70, which directly inform the base success probabilities $P_{\text{base}}(a)$ in our MDP transition model (Table 4). While BKT provides the student model, it does not supply a pedagogical policy; our work extends this perspective to a full sequential decision-making framework in which the tutoring agent actively selects actions to maximise cumulative learning outcomes. This reference underpins the transition probabilities of the main MDP but is not a competing system, and is therefore listed here as supporting background rather than in the main text.

D DQN Training Loss

Figure 14 reports the DQN training loss across the four seeds, referenced from the analysis in Section 5.2.

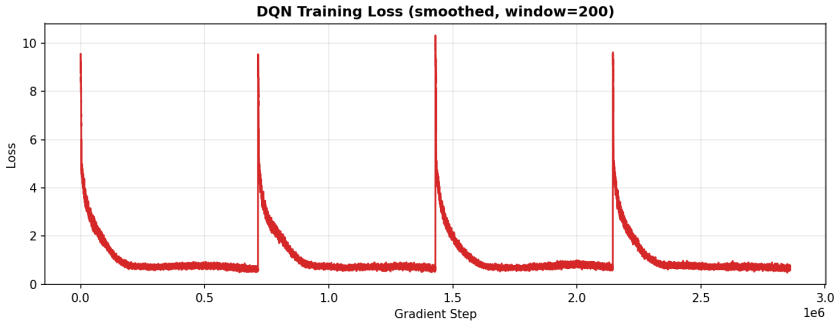

Figure 14: DQN training loss (Huber / Smooth L1, smoothed; four seeds concatenated). Each spike marks large-magnitude night-violation penalties (-30) entering the replay buffer, followed by a monotonic decay as the target network stabilises.