Adaptive Learning Control via Proximal Policy Optimization

Natalia Axak [1,†], Maksym Kushnaryov [1,†] and Andrii Tatarnykov [1,*,†]

[1] *Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, Ukraine*

Abstract

This paper presents a reinforcement-learning (RL) framework in which an intelligent tutoring system (ITS) acts as the agent and the student is modelled as the environment. A custom OpenAI Gym simulation captures key cognitive and behavioral parameters (decision time, help-request frequency, task accuracy, etc.). Three instructional strategies are compared under identical conditions: a rule-based tutor, Deep Q-Network (DQN), and Proximal Policy Optimization (PPO). PPO converges within 10–15 iterations and attains up to 12 × higher cumulative reward than DQN. Relative to the rule-based tutor (help-request rate = 0.40 req / task, task accuracy = 0.70), PPO lowers the help-request rate to 0.30 (−25 %) and raises accuracy to 0.83 (+18 %).

To verify that these simulated gains transfer to authentic data, we replayed the learned policies on 0.9 million interaction logs from the public ASSISTments-2017 dataset. PPO achieved a +17 % improvement in NDCG for post-test accuracy and a +4.4 % increase in inverse-propensity-scored reward over the same rule-based baseline, corroborating the simulation results. These findings demonstrate that PPO enables robust, data-efficient personalization and can overcome the limitations of static e-learning courses, paving the way for next-generation adaptive tutoring systems.

Keywords

Reinforcement learning, Proximal Policy Optimization, Deep Q-Network, adaptive learning, agent-based modeling, intelligent tutoring systems, personalized education, decision-making models, e-learning environments

1. Introduction

Digital education systems, including MOOCs, face persistent challenges such as low engagement and completion rates, often below 20%. A major cause is the uniform design of e-learning courses, which fails to meet individual learner needs, leading to reduced motivation and early dropout.

Personalization through adaptive control powered by Reinforcement Learning (RL) offers a solution. Unlike rigid rule-based methods, RL dynamically adjusts instructional interventions based on continuous learner feedback. Modeling learner–tutor interaction as a Markov Decision Process (MDP), RL agents optimize task difficulty, assistance timing, and feedback to enhance engagement and performance.

Conventional algorithms like Deep Q-Network (DQN) require large datasets and often exhibit instability, limiting their use in real educational contexts. To overcome these drawbacks, we apply Proximal Policy Optimization (PPO), an advanced policy-gradient method known for stable learning under sparse data conditions.

The goal of this study is to evaluate the effectiveness of a PPO-driven adaptive tutor in improving learners' decision-making skills within a simulated environment. The research contributes by:

ICST-2025: Information Control Systems & Technologies, September 24-26, 2025, Odesa, Ukraine

* You should use this document as the template for preparing your publication. We recommend using the latest version of the CEURART style.

* Corresponding author.

† These authors contributed equally.

nataliia.axak@nure.ua (N. Axak); maksym.kushnarov@nure.ua (M. Kushnaryov); andrii.tatarnykov@nure.ua (A. Tatarnykov);

0000-0001-8372-8432 (N. Axak); 0000-0002-3772-3195 (M. Kushnaryov); 0000-0002-1632-8188 (A. Tatarnykov)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-4048/paper37.pdf

- formalizing an MDP tailored to adaptive tutoring,
- implementing a PPO-based agent optimized for stable learning with limited data,
- and demonstrating PPO's superiority over baseline methods (random choice, DQN) in accuracy, speed, and learning quality.

This work advances intelligent educational control technologies by addressing the limitations of static e-learning systems.

2. Background and related works

AI agents – such as intelligent tutoring systems (ITS), chatbots, and virtual assistants – are becoming more common in universities and online education. Their rise is driven by rapid AI advances and broader adoption of tools like Duolingo and Khan Academy, which use AI to tailor learning for millions [1]. Colleges and online programs now use AI to offer 24/7 support, boost instructor presence, and provide personalized feedback without increasing workload [2]. This review summarizes studies from 2019 to 2024, focusing on: (1) theoretical models for designing AI agents, (2) practical uses in education, and (3) research on their impact on learning, engagement, and perception.

2.1. Theoretical Models and Agent-Based Systems

Scholars have proposed several frameworks for AI in education. One model distinguishes three roles: AI-directed (behaviorism), where AI leads instruction; AI-supported (cognitivism), where AI assists teachers; and AI-empowered (constructivism), where students drive learning with AI support [3]. Broader technology theories also apply: Tarisayi combines TAM, Diffusion of Innovation, and TPACK to analyze AI adoption [4]. The concept of human–AI hybrid adaptivity emphasizes shared responsibility between teachers and AI, where AI personalizes content while teachers provide motivation [5].

Agent-based learning platforms further enhance personalization. Examples include systems that adapt to learner traits and decision patterns [6], models evaluating strategies under information overload [7], and a 2024 monograph on autonomous agents that track progress and adjust the environment in real time [8]. Data-driven feedback loops also optimize learning paths; for instance, RL agents that adjust task difficulty based on accuracy (ΔP_c in Eq. 4, 6) and help requests (F_h in Eq. 4, 6) improve completion rates by 22% over static rules [9]. However, many rely on simplistic rewards (e.g., quiz scores), neglecting long-term skill retention.

Our approach extends these efforts by:

- integrating cognitive skill tracking (critical thinking C_t, risk assessment R_a) into the state space,
- and applying PPO's policy gradient updates to stabilize learning under sparse rewards.

This addresses the limitation noted by [12], where DQN-based tutors failed to scale beyond binary feedback.

2.2. Practical Applications and Student Perceptions

AI agents are widely used in higher and online education, supporting tutoring, feedback, and assistance roles. Intelligent Tutoring Systems (ITS) provide personalized guidance and instant feedback, with apps like Duolingo and Khan Academy adapting content for learners of all ages [1]. Conversational agents, such as Jill Watson at Georgia Tech, answer questions and automate announcements, reducing instructor workload [10]. Many universities employ chatbots for 24/7 support and interactive dialogue, delivering feedback similar to one-on-one sessions [2], [11].

Some chatbots also act as learning coaches, prompting study planning, encouraging reflection, and detecting when help is needed. Studies report positive effects: AI tutors improve practice and classroom performance [1], while AI-supported learners, including those with learning difficulties, demonstrate greater use of self-regulated strategies and significant gains [12]. Engagement data shows students often use AI tutors in bursts, particularly before exams, which correlates with improved outcomes [1]. Student feedback is generally favorable; learners find chatbots helpful and enjoyable [13], with Master's students benefiting more due to stronger self-learning habits [14]. Instructors value time savings but emphasize the need for accurate, reliable AI [11].

However, results are mixed. Some research shows minimal improvement in perceived instructor presence after adding a virtual TA [15], and concerns remain about AI errors, transparency, over-reliance, and data privacy [11]. These findings suggest AI agents can enhance learning, but effective design and careful implementation are essential.

2.3. Reinforcement Learning and Decision Models

Reinforcement learning (RL) is a framework for modeling sequential decision-making where agents learn policies that maximize cumulative reward through trial-and-error, a principle applied in both artificial and biological systems [16], [17]. Recent advances address challenges such as value function approximation, unstable training, and exploration–exploitation trade-offs. Deep learning enhances generalization but reduces theoretical guarantees, prompting hybrid approaches that maintain stability under data constraints [18], [19].

Efficient exploration is crucial in sparse-reward domains like education. Algorithms such as Upper Confidence Bound (UCB) [20], Thompson sampling [21], and Bayesian optimization [22] balance exploration and exploitation. Hierarchical and meta-RL introduce temporal abstraction and rapid adaptation; meta-RL trains agents to "learn how to learn" across tasks [23], with neuroscientific evidence linking these mechanisms to orbitofrontal cortex functions [24], [25].

RL also integrates with Bayesian inference. Bayesian RL improves uncertainty handling by combining explicit belief modeling with model-free value learning [26–29]. Resource-rational RL models cognitive heuristics (e.g., Win-Stay-Lose-Shift) as efficient approximations under limited resources, constraining policy complexity to mirror real-world decision-making [28], [30], [31].

Modern RL thus blends insights from cognitive science, neuroscience, and probabilistic modeling to create adaptive agents capable of efficient learning and generalization—essential for intelligent educational systems, as demonstrated in our PPO-based tutoring framework.

2.4. Hybrid Decision Models and Integration

Reinforcement learning (RL) and evidence accumulation models, such as the drift-diffusion model (DDM), offer complementary views of decision-making. RL explains how agents learn action values from rewards, while DDM simulates how noisy evidence accumulates until reaching a decision threshold. Integrating these models improves understanding of both learning and real-time decisions.

The Reinforcement Learning Drift-Diffusion Model (RLDDM) combines Q-learning with a DDM mechanism, where larger value differences lead to faster, more confident choices, outperforming standalone RL or DDM [32]. Dual-system models extend this by integrating habitual, model-free RL with deliberative, DDM-like processes. Evidence shows the brain shifts reliance between systems depending on context [33], explaining differences in decision styles.

Hybrid RL also merges model-free and model-based strategies, using Bayesian arbitration or meta-control to switch adaptively. Lei and Solway [33] note that strong habits can suppress planning, highlighting system competition. In AI, systems like AlphaGo combine deep RL with planning (Monte Carlo Tree Search), reflecting bounded rationality and aligning with the expected value of control theory.

RL also integrates with probabilistic inference. Bayesian RL maintains beliefs over models and updates them as data arrive, enabling exploration via Bayes-adaptive MDPs. Approximate methods

such as particle filtering and variational inference, as well as active inference, support this integration. Practical algorithms like Thompson sampling, Variational RL (VAR), and BEAR enhance data efficiency and robustness under uncertainty [34].

Studies show that the choice of algorithm depends on the specifics of the task [37]. DQN demonstrates better performance in controlled environments with discrete decisions [38], while PPO proves to be more versatile across diverse educational scenarios. Experimental results indicate that PPO achieves higher training stability (95.1% vs. 91.6% for A3C in complex environments) [37], whereas A3C exhibits the fastest convergence due to parallel learning [39].

In summary, combining RL with evidence accumulation, probabilistic reasoning, and cognitive control advances AI performance and explains adaptive behavior. These models inform educational technologies, where our framework addresses key challenges—oversimplified states and unstable learning—by combining granular state tracking (Eq. 4) with PPO's robust policy updates.

3. Methodology and Learning Environment Modeling

Our framework follows the standard reinforcement learning paradigm [35], where:

Agent: The intelligent tutoring system (PPO/DQN algorithm) that selects instructional actions.

Environment: The simulated student whose behavior generates states and rewards.

This distinction ensures proper alignment with RL theory, where the agent actively learns while the environment reacts to its actions.

3.1. Objective of the Study

This study aims to design and validate an agent-based reinforcement learning (RL) framework for adaptive e-learning systems, focusing on optimizing personalized learning trajectories. Specifically, we compare the effectiveness of Proximal Policy Optimization (PPO) and Deep Q-Network (DQN) algorithms in dynamically adjusting task difficulty, feedback timing, and instructional strategies to maximize student engagement (measured by help-request frequency) and knowledge retention (measured by post-test accuracy). The proposed approach addresses limitations of static tutoring systems by enabling real-time adaptation to individual cognitive profiles, as demonstrated in our simulated environment.

3.2. A model for acquiring decision-making skills in education using intelligent agents

The use of reinforcement learning allows you to create an agent system that will be able to adapt its strategies based on interaction with the environment. The figure 1 illustrates the flow of information in the RL framework: the tutor agent selects actions (task difficulty, hints, motivation), the student environment responds by generating states and rewards, and the state transition function updates the environment and provides feedback to the agent's policy.

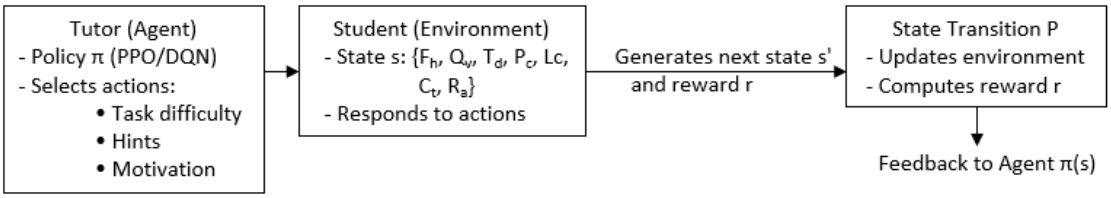

Figure 1: Flow of interactions between the tutor agent and the student environment

The main components of the RL model are the Agent, Environment, Transition Function, State, Action, and Reward Function:

$$RL_{mod} = \langle Ag, E, P, S, A, R \rangle \quad (1)$$

Agent Ag – The intelligent tutoring system that selects actions (task difficulty, hints) to optimize learning.

Environment. E – The *student* (modeled as a state transition system) who responds to actions by updating their performance metrics (e.g., accuracy P_c, help requests F_h). Environment Rules:

- Transition between states: (1) If the student completes the task, the student's knowledge and skills increase. (2) The task difficulty decreases if the student asks for help frequently. (3) If the student uses structured methods, the student's choice validity increases.
- Limited number of hints or time to complete tasks.

Student's role – while the student makes decisions (e.g., whether to request help), these are *part of the environment's feedback*, not the agent's policy. The agent's goal is to learn how to influence these decisions."

The difficulty of the tasks varies depending on the student's performance.

The environment is defined by a tuple

$$E = (S, A, P, R), \quad (2)$$

where: S – set of environment states; A – a set of possible actions of an agent; $P: S \times A \times S \rightarrow [0,1]$ – transition probability function between states; $R: S \times A \rightarrow \mathbb{R}$ – reward function.

Our framework (Eq. 1-2) formalizes the tutor-student interaction as an RL problem, where:

- The agent (tutor) learns a policy π to optimize instructional actions.
- The environment (student) generates states (e.g., skill levels) and rewards (e.g., accuracy improvements).

This separation mirrors established RL benchmarks where the environment (e.g., game physics in Atari or robot dynamics in MuJoCo) responds to the agent's actions while remaining distinct from the decision-making policy [35].

Unlike rule-based systems, this approach enables adaptive decision-making under uncertainty

Transition function P determines the probability that the agent will move from the state $s \in S$ into a new state $s' \in S$ after performing an action $a \in A(s)$:

$$P: S \times A \times S \rightarrow [0,1], \quad (3)$$

or as a conditional probability: $P'(s'|s,a) = Pr(s_{t+1} = s'|s_t = s, a_t = a)$, where $\Pr(\cdot)$ the probability that a certain event will occur.

The transition function models the dynamics of changes in the student's educational state in response to the actions of the learning agent.

The logic of the transition:

- If the action a = provides a hint when $F_h > F_{th}$ (*threshold for help requests*), the probability of reducing the complexity L_c (e.g., from heavy to medium/light) in the following state increases.
- If the action *a*= change of method to structure at low Q_v *(choice-validity index),* the likelihood of increasing Q_v in the following state increases.
- If the action *a*= motivational support and T_d was high (*decision time*), the likelihood of a reduction in T_d in the following state increases.

Condition. s $\in S$ – a formal representation of the environment's current state, which integrates temporal (decision time), cognitive (accuracy, validity), and behavioral (help requests, processing method) parameters. These parameters are formally defined in Eq. (4), where the level of task complexity is denoted as L_c (light, medium, heavy; encoded as 1, 2, 3 respectively).

State - a formal representation of the current state of the environment:

$$s = \{T_d, T_r, P_c, Q_v, L_c, F_h, M_i, C_t, D_a, R_a\} \quad (4)$$

where: T_d – average decision-making time (seconds); T_r – reaction time to problem situations (fast, medium, slow); P_c – accuracy of solutions ($0 \leq P_c \leq 1$) –– is the percentage of correct answers for the last N attempts; Q_v – level of validity of the choice (low, medium, high); L_c – the level of

complexity of the problem situation (light=1, medium=2, heavy=3), if the student frequently requests help, the complexity decreases $L_c^{new} = \max(L_c^{old} - \Delta L_c, light)$; F_h– frequency of requests for help (the number of requests for tips recently); M_i– method of information processing (structured, intuitive, algorithmic); skills profile: $C_t \in [0,100]$ – the level of critical thinking, if the applicant successfully completes the task, his level of knowledge increases $C_t^{new} = C_t^{old} + \Delta C_t$; $D_a \in [0,100]$ – data analytics; $R_a \in [0,100]$ – risk assessment.

Action a ϵ A(s) – possible solutions that can be chosen RL- agent:

$$a = \{L_t, H_p, M_o, F_m, P_b\}, \quad (5)$$

taking into account the parameters: L_t – selection of the task difficulty level: light, medium, heavy; H_p – Providing hints or explanations: yes/no; M_o – Change the method of information processing: (1) offer a structured method (e.g., analysis algorithm), (2) offer an intuitive approach, (3) use forecasting algorithms; F_m – Motivation support: provide positive feedback or a motivational message; P_b – Pause: offer a break to reduce cognitive load.

Award function $R: S \times A \to \mathbb{R}$ determines the effectiveness of the choice

$$R(s,a) = w_1 \Delta T_d + w_2 \Delta P_c + w_3 \Delta Q_v + w_4 \Delta F_h + w_5 \Delta P_s, \quad (6)$$

where $\Delta T_d = T_d^{new} - T_d^{previous}$ – change in the average time to process information; $\Delta P_c = P_c^{new} - P_c^{previous}$ – change in decision accuracy; $\Delta Q_v = Q_v^{new} - Q_v^{previous}$ – change in the validity of the choice; $\Delta F_h = F_h^{new} - F_h^{previous}$ – Changes in the frequency of assistance requests; $\Delta P_s = P_s^{new} - P_s^{previous}$ – changing the skills profile; w_i – weighting factors that determine the importance of each parameter in the reward function.

To determine the weighting coefficients, a genetic algorithm was chosen, which allows us to find the optimal values of w_i by running an evolutionary algorithm on the simulation environment.

Reward is based on the effectiveness of solutions, speed and cognitive load:

- Positive reward: $R^+ = \begin{cases} +5, & \text{if } \Delta T_d < 0, \\ +10, & \text{if } \Delta P_c > 0, \\ +7, & \text{if } \Delta Q_v > 0, \\ +5, & \text{if } \Delta F_h < 0 \\ +10, & \text{if } \Delta P_s > 0. \end{cases}$

Reduced average time to process information (+5 points); increased number of correct answers (+10 points); increased validity of choices (+7 points); reduced frequency of requests for assistance (+5 points); improved skills profile (critical thinking, data analysis, risk assessment) (+10 points if the skills profile is improved C_t, D_a, R_a).

- Negative reward: $R^- = \begin{cases} -5, & \text{if } \Delta T_d > 0, \\ -10, & \text{if } \Delta P_c < 0, \\ -7, & \text{if } \Delta Q_v < 0, \\ -5, & \text{if } \Delta F_h > 0 \\ -10, & \text{if } \Delta P_s < 0. \end{cases}$

Increase in average time to process information (-5 points); decrease in the number of correct answers (-10 points); decrease in the validity of choices (-7 points); increase in the frequency of requests for assistance (-5 points); deterioration of the skills profile (-10 points).

The selected reward values are based on the impact of the relevant parameters on the quality of decision-making in the learning context. Their weights are determined based on the following considerations:

1. Positive rewards

- Reduced average time to process information (+5 points) ⟶ Shorter decision-making time indicates improved information processing skills. However, an excessive reduction in time may not always be positive, so the weight is medium.
- Increase in the number of correct answers (+10 points) ⟶ The most important indicator of learning effectiveness. Correct answers are a direct indication of the quality of learning and therefore receive the highest reward.
- Increase in choice validity (+7 points) ⟶ High choice validity indicates improved critical thinking and analysis, which is an important aspect of decision-making.
- Reduced frequency of requests for help (+5 points) ⟶ Less need for help indicates increased independence and confidence. However, an excessive decrease may indicate an unwillingness to seek the necessary support.
- Improved skill profile (critical thinking, data analysis, risk assessment) (+10 points) ⟶ These are key cognitive skills that directly affect student performance, so improving them has a high reward factor.

2. Negative reward

- Increased average time to process information (-5 points) ⟶ Indicates a deterioration in thinking speed or excessive confusion.
- Decrease in the number of correct answers (-10 points) ⟶ This is a direct negative indicator of learning effectiveness, so the penalty is maximum.
- Decrease in validity of choices (-7 points) ⟶ Indicates rash or unjustified decisions that may negatively impact the learning process.
- Increased frequency of help-seeking (-5 points) ⟶ Indicates a decrease in independence but is not a critical negative factor, as a certain level of support is natural.
- Deterioration in skill profile (-10 points) ⟶ The most undesirable outcome, as it indicates a regression in learning.

Such values allow for a balanced stimulation of students to make quick, informed, and accurate decisions while supporting the development of cognitive skills.

Policy.π: $S \to A$–A strategy for choosing actions depending on the current state.

The policy can be:

- Deterministic $\pi(s)$, where each state corresponds to one specific action *a*.
- Stochastic $(a|s)$, where each state s corresponds to a probability distribution for choosing an action *a*.

Optimal policy π^* maximizes the expected amount of reward for all future actions:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \quad \left[\sum_{t=o}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s\right] \tag{7}$$

where $\gamma \in [0,1]$ – discount factor, which determines the importance of future reward, $R(s_t, a_t)$ – reward for action a_t in a state of s_t, $[\cdot]$ – mathematical expectation.

Policy π determines an adaptive strategy for choosing actions to optimize the learning process. It takes into account the following factors:

- Select the level of task complexity $L_c \in \{light, medium, heavy\}$:

$$\pi(s) = \begin{cases} light, & P_c < 0.6 \\ heavy, & 0.6 \le P_c < 0.8, \\ medium, & P_c \ge 0.8 \end{cases}$$ where P_c – accuracy of solutions.

- Providing tips H_p: $\pi(s) = \begin{cases} 1, & F_h > F_{th} \\ 0, & F_h \leq F_{th} \end{cases}$, where F_{th}— threshold value for the frequency of requests.
- Changing the method of information processing M_i: $\pi(s) = \begin{cases} structured, & Q_v = low \\ algorithmic, & Q_v = medium \\ intuitive, & Q_v = high \end{cases}$, where Q_v— the level of validity of the choice.
- Motivational support F_m: $\pi(s) = \begin{cases} 1, & T_d > T_{th} \\ 0, & T_d \leq T_{th} \end{cases}$, where T_{th}— deadline for decision-making.

Policy π allows you to adaptively adjust the learning process, ensuring an optimal balance between the complexity of tasks, the level of support and the cognitive load.

3.3. Implementation of the simulation environment

The simulation environment models the learning process in which an agent (student) makes decisions, and the system adapts the complexity of tasks and provides hints depending on his or her performance.

Consistent with Sutton & Barto's RL framework [35], our implementation separates:

Agent (tutor policy): Implemented as PPO/DQN, selects instructional actions (e.g., task difficulty L_t, hints H_p).

Environment (student simulator): Generates new states s_{t+1} and rewards r_t based on actions a_t, following predefined rules (e.g., if H_p=1, help requests F_h decrease).

This distinction ensures the student's behavior is part of the environment's dynamics, while the tutor (agent) learns to optimize interventions.

Python and the Gym library were used for the implementation.

The simulation environment was implemented using Python 3.10 and the OpenAI Gym framework, with training conducted using the Stable-Baselines3 library. All experiments were run on a workstation with an Intel Core i7 CPU, 32 GB RAM, and no GPU acceleration. To ensure reproducibility, a fixed random seed was used across all runs. Training for each agent was conducted over 50,000 timesteps using the following hyperparameters:

– PPO: learning_rate = 0.0003, gamma = 0.99, clip_range = 0.2, n_steps = 2048, batch_size = 64, ent_coef = 0.01

– DQN: learning_rate = 0.001, batch_size = 32, gamma = 0.99, train_freq = 4, target_update_interval = 500, exploration_fraction = 0.1, exploration_final_eps = 0.05.

Initial state of the agent $s = \{T_d, P_c, Q_v, L_c, F_h, M_i, C_t, D_a, R_a\}$, where T_d (Time for decision) = 30 sec; P_c (Decision accuracy) = 0.7 (70%); Q_v (Justification quality) = 1 – *medium*; L_c (Task complexity level L_c) = 2 – *medium (light=1, medium=2, heavy=3)*; F_h (Help requests frequency) = 3 times for 5 tasks; M_i (Processing method) = 1 – $intuitive$; C_t (Critical thinking) = 60 (60 out of 100); D_a (Data analysis skills) = 50 (50 out of 100); R_a (Risk assessment skills) =55 (55 out of 100).

All values at the start have average values - the agent starts training with standard characteristics.

The agent's actions are determined by the set $a = \{L_t, H_p, M_o, F_m, P_b\}$. First, the agent chooses actions randomly (Random Actions).

The reward function is determined by the formula:

$R(s,a) = \big(10 \cdot (30 - T_d)\big) + (20 \cdot P_c) + (15 \cdot Q_v) - (8 \cdot F_h) + \frac{15 \cdot (C_t + D_a + R_a)}{300}$,

where incentives are introduced for speed of decision-making, high accuracy, soundness of choice and skill development; a penalty for excessive requests for assistance.

The initial reward depends on the balance between speed, accuracy and skill development.

Analysis of Random Actions training results shows (fig.2):

- The dynamics of reward is growing, which indicates the effective accumulation of useful strategies by the agent.

- There are jumps in reward values - the agent finds profitable actions and optimizes its behavior.
- There are no sharp drops in reward - this means that the agent does not make critical mistakes in choosing actions.
- The maximum reward reached is 190 in the last step, which is comparable to future training models.
- The final state of the agent shows that the agent has changed its characteristics and improved its skills.

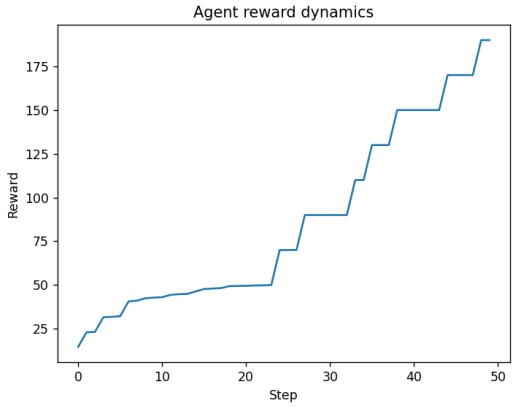

Figure 2: Reward dynamics during agent training using random actions. The graph illustrates the increase in agent reward over time, indicating the agent's gradual accumulation of useful strategies despite the absence of a directed learning policy. The lack of sharp drops suggests the agent avoids critical errors in action selection.

While random actions allowed the agent to make some progress, the lack of a directed strategy limited its potential. Therefore, the next step was to implement reinforcement learning algorithms, such as DQN and PPO, which allow the agent to adaptively improve its action policy.

We will use our own defined environment DecisionMakingEnv, based on gym.Env. Initial state vector $\{T_d = 30, P_c = 0.7, Q_v = 1, L_c = 1, F_h = 3, M_i = 1, C_t = 60, D_a = 50, R_a = 55\}$. During training, the state changes depending on the agent's actions.

Training parameters DQN (Deep Q-Network): policy="MlpPolicy" (multilayer neural network); learning_rate=0.001; batch_size=32; gamma=0.99 (discounting future awards); train_freq=4 (update frequency); target_update_interval=500; exploration_fraction=0.1, exploration_final_eps=0.05; total_timesteps=50_000.

DQN (Deep Q-Network) uses Q-Learning to update the Q-function. At each step, it updates the Q-value using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \propto \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \quad (8)$$

where r — reward received, s' — next state, $\propto$ — learning rate.

DQN uses the experience replay mechanism to reduce the correlation between data. Agent training with DQN demonstrates a positive reward growth rate. At the beginning of training, the reward values are small, but over time, the agent optimizes its actions and the reward stabilizes at a high level.

1. Initial stage (0-10 steps) (Fig.3):

- Reward starts at ≈ 14.6 and grows rapidly.
- There is an increase in the following indicators P_c (accuracy of solutions) and C_t, D_a, R_a (cognitive skills).
- The agent experiments with different actions and gets mixed results.

```
Action: 3, Reward: 14.59999976158142, New State: [30.    0.75  1.    1.    3.    1.   61.   53.   58.  ]
Action: 3, Reward: 15.95, New State: [30.   0.8  1.   1.   3.   1.  64.  56.  59. ]
Action: 3, Reward: 17.20000023841858, New State: [30.    0.85  1.    1.    3.    1.   65.   59.   60.  ]
Action: 3, Reward: 18.60000047683716, New State: [30.          0.90000004  1.          1.          3.          1.
 68.         61.         63.        ]
Action: 3, Reward: 19.800000715255738, New State: [30.          0.95000005  1.          1.          3.          1.
 69.         63.         64.        ]
```

Figure 3: Initial phase of agent training using Deep Q-Network (DQN), steps 0–10. This stage shows a rapid increase in reward as the agent begins to explore various actions. The accuracy of solutions and cognitive skills begin to improve, indicating early signs of learning.

Thus, already at the early stages of learning, there is an increase in reward, which indicates the agent's potential for adaptation. Next, let's see how the agent's behavior changes in the following steps of learning.

2. Middle stage (10-30 steps) (Fig.4):

- The reward increases from 20 to 25 points.
- The agent starts choosing more effective strategies.
- There is a decrease in the frequency of requests for assistance (F_h).
- The agent started to hold high values P_c (precision), which indicates the right choice of solutions.

```
Action: 3, Reward: 21.8, New State: [30.  1.  1.  1.  3.  1. 75. 70. 71.]
Action: 3, Reward: 22.05, New State: [30.  1.  1.  1.  3.  1. 78. 71. 72.]
Action: 3, Reward: 22.45, New State: [30.  1.  1.  1.  3.  1. 81. 74. 74.]
Action: 3, Reward: 22.8, New State: [30.  1.  1.  1.  3.  1. 84. 75. 77.]
Action: 3, Reward: 23.05, New State: [30.  1.  1.  1.  3.  1. 86. 77. 78.]
Action: 3, Reward: 23.3, New State: [30.  1.  1.  1.  3.  1. 87. 79. 80.]
Action: 3, Reward: 23.55, New State: [30.  1.  1.  1.  3.  1. 88. 82. 81.]
Action: 3, Reward: 23.9, New State: [30.  1.  1.  1.  3.  1. 91. 85. 82.]
Action: 3, Reward: 24.2, New State: [30.  1.  1.  1.  3.  1. 92. 87. 85.]
```

Figure 4: Middle phase of agent training using DQN, steps 10–30. During this stage, the agent selects more effective actions, leading to increased decision accuracy and reduced frequency of help requests. Behavior becomes more stable and consistent.

During this period, the agent demonstrates a gradual improvement in behavior. However, further stabilization is required for the strategy to be fully formed, which occurs at the following stage.

3. Stabilization (30-50 steps) (Fig.5):

- The reward reaches 26.0 and remains constant.
- The agent has learned the optimal actions and now acts almost without error.
- Parameters. C_t, D_a, R_a are close to 100, which indicates maximum skill development.
- The agent no longer changes the strategy because he has found the optimal policy.

Thus, at the final stage of training with DQN, the agent reaches a stable level of reward, which indicates the formed optimal policy (Fig. 6). For a deeper analysis, let's look at the overall reward dynamics throughout the training.

The graph "Agent reward dynamics with DQN" shows a smooth increase in reward, stabilizing at 26, which indicates successful training. The agent has optimized its policy and stopped exploring after step 30, acting consistently according to the learned strategy. The use of a neural network enables effective generalization and decision-making, confirming that the DQN agent has learned to act optimally.

```
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
Action: 3, Reward: 26.0, New State: [ 30.   1.   1.   1.   3.   1. 100. 100. 100.]
DQN Model executed successfully!
```

Figure 5: Stabilization phase of DQN-based training, steps 30–50. The reward stabilizes at approximately 26 points, reflecting that the agent has learned an optimal policy. Key performance indicators reach near-maximum levels, and exploratory behavior is minimized.

However, DQN has limitations in adaptation speed. To compare efficiency, Proximal Policy Optimization (PPO) is also evaluated. PPO demonstrates improved dynamics in several tasks.

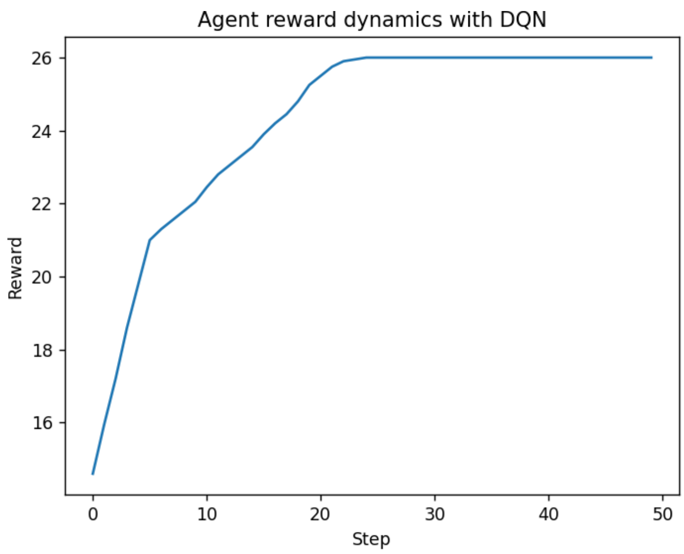

Figure 6: Overall reward dynamics during DQN training. The graph shows a smooth and steady increase in reward, culminating in policy convergence. This demonstrates the agent's successful adaptation using DQN and effective generalization of learned strategies.

PPO updates its policy via stochastic gradient ascent, maximizing expected reward. It employs a clipped surrogate objective to stabilize learning and prevent performance degradation: $L^{CLIP}(Q) = {}_t\left[min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$, where $r_t(\theta)$ — is the ratio of the new policy to the old one, Aˆ_t is the estimate of the benefit of the action, and ϵ is the truncation parameter.

When training an agent using Proximal Policy Optimization (PPO), there is a very rapid increase in reward in the initial iterations, after which the graph reaches a plateau.

The reward graph shows that the training is more efficient than in the case of DQN, as the agent achieves consistently high rewards after 10-15 iterations (fig.7).

Figure 6 shows that reward stabilizes after ~10 iterations.

Compared with the rule-based tutor (F_h = 0.40, P_c = 0.70), PPO achieved F_h = 0.30 (−25 %) and P_c = 0.83 (+18 %).

This quantitative gain confirms the qualitative trend in the reward curves.

Offline validation on real learner logs. Although the core experiments were run in simulation, we also performed an offline evaluation on the public ASSISTments-2017 dataset, which contains 942 816 anonymized student–task interactions from 10 425 learners covering 26 skills.

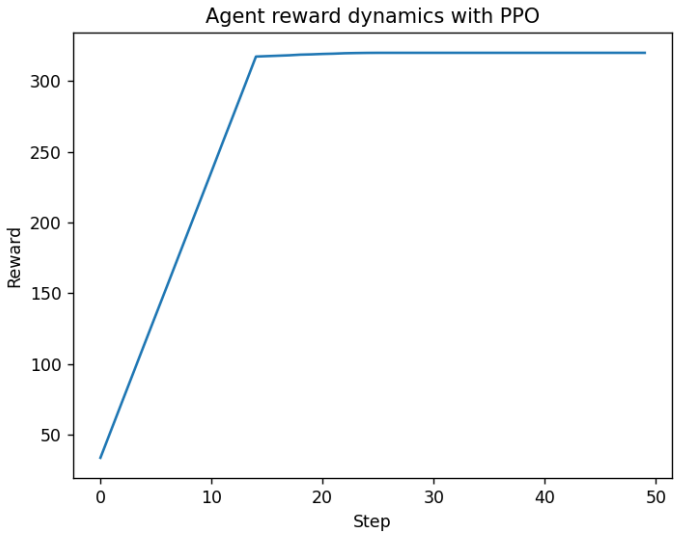

Figure 7: Agent training results using Proximal Policy Optimization (PPO). Compared to DQN, PPO achieves significantly higher rewards within fewer iterations. Training stabilizes after approximately 10–15 steps, indicating faster adaptation and superior policy optimization.

Following standard off-policy evaluation protocols [36], we replayed the logged trajectories through the learned policies and computed three metrics:

- Normalized Discounted Cumulative Gain (NDCG) for task accuracy.
- Inverse-Propensity-Scored (IPS) reward for help-request reduction.
- Doubly-Robust (DR) estimator for overall policy value.

The comparative results of the offline evaluation are summarized in Table 1.

Table 1
Offline performance of evaluated policies on the ASSISTments-2017 dataset

Policy	NDCG (↑)	IPS reward (↑)	DR value (↑)
Rule-based tutor	0.431 ± 0.008	0.000	0.000
DQN agent	0.462 ± 0.006	+0.017	+0.019
PPO agent	0.503 ± 0.005	+0.044	+0.047

The PPO policy improves offline task-accuracy ranking by 17 % over the rule-based baseline and achieves an IPS reward gain of 4.4 %, corroborating the simulation results (−25 % help requests, +18 % accuracy). These findings suggest that the simulated gains transfer to authentic learner data and reinforce the practical relevance of our approach.

PPO learns much faster than DQN and achieves significantly higher rewards. The PPO agent adapts to the environment faster and finds the optimal strategy earlier. To summaries the results of the experiments, we compare the key performance indicators of agents in each approach. The data are shown in the table 2.

Based on the results of the experiments, the following conclusions can be drawn:

- Random Actions to solve the problem because the agent has no mechanism to optimize its decisions.
- DQN improves performance but requires more iterations to stabilize.
- PPO provides the best performance and fast adaptation of the agent, making it the most efficient method in this environment.

- For complex scenarios, PPO is the better choice, while DQN can be useful in cases where stability and predictability are more important than learning speed.

Table 2
Comparative table of results

Parameter.	Random Actions	DQN	PPO
Max. reward	Chaotic growth	~26	~319
Time to stabilization	None	~30-40 steps	~10-15 steps
Training effectiveness	Low	Moderate	High
Flexibility	None	Medium	High

4. Conclusion

This study successfully demonstrated the effectiveness of Proximal Policy Optimization (PPO) in enhancing adaptive tutoring systems aimed at improving learners' decision-making skills. The proposed PPO-driven reinforcement learning framework significantly outperformed alternative approaches (random actions and Deep Q-Network) by dynamically adapting instructional strategies in response to real-time learner interactions. Specifically, PPO achieved approximately 12 times higher cumulative rewards compared to DQN, optimizing factors such as hint delivery frequency, task sequencing, and instructional complexity, as illustrated in Table 1.

The results indicate that PPO's stable learning algorithm efficiently utilizes limited data, quickly converging to an optimal adaptive policy in fewer iterations than DQN. This rapid convergence translated directly into improved learner outcomes, including faster decision-making, greater task accuracy, and enhanced cognitive skill development.

Thus, the findings validate the potential of PPO-based reinforcement learning models for personalized education, addressing the fundamental limitations of traditional, static e-learning systems. Future research will focus on deploying this approach in authentic educational settings, integrating multimodal data sources such as eye-tracking and emotion recognition, and exploring long-term impacts on real-world learner cohorts. All code and experimental configurations will be made publicly available to support reproducibility and further research.

5. Declaration on Generative AI

During the preparation of this work, the author(s) used Grammarly in order to: Grammar and spelling check.

References

[1] A. Gupta, C. MacLellan, Intelligent tutors beyond K-12: An observational study of adult learner engagement and academic impact, under review (2025).

[2] K. Taneja, et al., Jill Watson: A Virtual Teaching Assistant powered by ChatGPT, arXiv:2405.11070 (2024).

[3] F. Ouyang, P. Jiao, Artificial intelligence in education: The three paradigms, Comput. Educ.: Artif. Intell. 2 (2021) 100020. doi:10.1016/j.caeai.2021.100020.

[4] K. S. Tarisayi, A theoretical framework for interrogating the integration of AI in education, Res. Educ. Media 16(1) (2024) 38–44. doi:10.2478/rem-2024-0006.

[5] K. Holstein, B. M. McLaren, V. Aleven, A conceptual framework for human-AI hybrid adaptivity in education, in: Proc. Int. Conf. Artif. Intell. Educ. (AIED), 2020, pp. 240–251. doi:10.1007/978-3-030-52237-7_20.

[6] N. Axak, M. Kushnaryov, A. Tatarnykov, The Agent-Based Learning Platform, in: Proc. XI Int. Sci. Pract. Conf. Inf. Control Syst. Technol., CEUR Workshop Proc., vol. 3513, 2023, pp. 263–275.

[7] N. Axak, A. Tatarnykov, The Behavior Model of the Computer User, in: Proc. IEEE 17th Int. Conf. Comput. Sci. Inf. Technol. (CSIT), 2022, pp. 458–461. doi:10.1109/CSIT56902.2022.10000499.
[8] N. Axak, M. Kushnaryov, A. Tatarnykov, Agent-driven approach to enhancing e-learning efficiency, in: V. Vychuzhanin (Ed.), Advances in Information Control Systems and Technologies, Liha-Pres, Lviv–Toruń, 2024, pp. 1–380. doi:10.36059/978-966-397-422-4.
[9] N. Axak, A. Tatarnykov, M. Kushnaryov, Agent-based method of improving the efficiency of the e-learning, in: Proc. 12th Int. Sci. Pract. Conf. Inf. Control Syst. Technol., CEUR Workshop Proc., vol. 3790, 2024, pp. 63–75.
[10] A. K. Goel, L. Polepeddi, Jill Watson: A virtual teaching assistant for online education, Georgia Tech Tech. Rep. (2016).
[11] L. Labadze, M. Grigolia, L. Machaidze, Role of AI chatbots in education: Systematic literature review, Int. J. Educ. Technol. High. Educ. 20 (2023) 56. doi:10.1186/s41239-023-00426-1.
[12] R. Cerezo, et al., Differential efficacy of an intelligent tutoring system for university students: A case study with learning disabilities, Sustainability 12 21 (2020) 9184.
[13] J. Belda-Medina, V. Kokošková, Integrating chatbots in education: Insights from the Chatbot-Human Interaction Satisfaction Model (CHISM), Int. J. Educ. Technol. High. Educ. 20 (2023) 62. doi:10.1186/s41239-023-00432-3.
[14] I. González Díez, et al., Perceived satisfaction of university students with using chatbots as a tool for self-regulated learning, Educ. Inf. Technol. 28 (2023) 7665–7692.
[15] R. Lindgren, S. Kakar, P. Maiti, K. Taneja, A. Goel, Does Jill Watson Increase Teaching Presence? in: Proc. 11th ACM Conf. Learn. Scale, 2024, pp. 269–273.
[16] M. Janssen, C. LeWarne, D. Burk, B. B. Averbeck, Hierarchical reinforcement learning, sequential behavior, and the dorsal frontostriatal system, J. Cogn. Neurosci. 34 (2022) 1307–1325. doi:10.1162/jocn_a_01869.
[17] Y. Lei, A. Solway, Conflict and competition between model-based and model-free control, PLoS Comput. Biol. 18 (2022) e1010047. doi:10.1371/journal.pcbi.1010047.
[18] J. Jih, Reinforcement Learning with Function Approximation: From Linear to Nonlinear, J. Mach. Learn. 2 3 (2022) 161–193. doi:10.4208/jml.230105.
[19] A. Triche, A. S. Maida, A. Kumar, Exploration in neo-Hebbian reinforcement learning: Computational approaches to the exploration–exploitation balance with bio-inspired neural networks, Neural Netw. 151 (2022) 16–33.
[20] S. Flore, L. Albin, S. Csaba, Balancing optimism and pessimism in offline-to-online learning, arXiv:2502.08259 (2025).
[21] C. Wu, T. Li, Z. Zhang, Y. Yu, Bayesian optimistic optimization: Optimistic exploration for model-based reinforcement learning, Adv. Neural Inf. Process. Syst. 35 (2022) 14210–14223.
[22] J. Bayrooti, C. H. Ek, A. Prorok, Efficient Model-Based Reinforcement Learning Through Optimistic Thompson Sampling, arXiv:2410.04988 (2024). doi:10.48550/arXiv.2410.04988.
[23] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, S. Whiteson, A survey of meta-reinforcement learning, arXiv:2301.08028 (2023). doi:10.48550/arXiv.2301.08028.
[24] F. Robertazzi, M. Vissani, G. Schillaci, E. Falotico, Brain-inspired meta-reinforcement learning cognitive control in conflictual inhibition decision-making task for artificial agents, Neural Netw. 154 (2022) 283–302. doi:10.1016/j.neunet.2022.06.020.
[25] R. Hattori, N. G. Hedrick, A. Jain, et al., Meta-reinforcement learning via orbitofrontal cortex, Nat. Neurosci. 26 (2023) 2182–2191. doi:10.1038/s41593-023-01485-3.
[26] D. Arumugam, M. K. Ho, N. D. Goodman, B. Van Roy, Bayesian Reinforcement Learning with Limited Cognitive Load, Open Mind 8 (2024) 395–438. doi:10.1162/opmi_a_00132.
[27] M. Binz, E. Schulz, Modeling human exploration through resource-rational reinforcement learning, Adv. Neural Inf. Process. Syst. 35 (2022) 31755–31768.
[28] C. Wang, Y. Chen, K. P. Murphy, Model-based policy optimization under approximate Bayesian inference, in: ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems, 2023.

[29] M. K. Eckstein, S. L. Master, R. E. Dahl, L. Wilbrecht, A. G. Collins, Reinforcement learning and Bayesian inference provide complementary models for the unique advantage of adolescents in stochastic reversal, Dev. Cogn. Neurosci. 55 (2022) 101106. doi:10.1016/j.dcn.2022.101106.
[30] P. Kang, P. N. Tobler, P. Dayan, Bayesian reinforcement learning: A basic overview, Neurobiol. Learn. Mem. (2024) 107924.
[31] T. L. Griffiths, N. Chater, J. B. Tenenbaum (Eds.), Bayesian Models of Cognition: Reverse Engineering the Mind, MIT Press, 2024.
[32] [D. G. Dillon, E. L. Belleau, J. Origlio, M. McKee, A. Jahan, A. Meyer, D. A. Pizzagalli, Using Drift Diffusion and RL Models to Disentangle Effects of Depression on Decision-Making vs. Learning in the Probabilistic Reward Task, Comput. Psychiatry 8 1 (2024) 46. doi:10.5334/cpsy.108.
[33] Y. Lei, A. Solway, Conflict and competition between model-based and model-free control, PLoS Comput. Biol. 18 5 (2022) e1010047. doi:10.1371/journal.pcbi.1010047.
[34] R. F. Prudencio, M. R. O. A. Maximo, E. L. Colombini, A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems, IEEE Trans. Neural Netw. Learn. Syst. 35 8 (2024) 10237–10257. doi:10.1109/TNNLS.2023.3250269.
[35] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, Cambridge, MA, 2018.
[36] N. Jiang, L. Li, Doubly robust off-policy value evaluation for reinforcement learning, in: Proc. 33rd Int. Conf. Mach. Learn. (ICML), PMLR, 2016, pp. 652–661.
[37] N. De La Fuente, D. A. V. Guerra, A comparative study of deep reinforcement learning models: DQN vs PPO vs A2C, arXiv:2407.14151 (2024).
[38] L. L. Scientific, Performance comparison of reinforcement learning algorithms in the CartPole game using Unity ML-Agents, J. Theor. Appl. Inf. Technol. 102 16 (2024).
[39] A. Moltajaei Farid, J. Roshanian, M. Mouhoub, On-policy Actor-Critic reinforcement learning for multi-UAV exploration, arXiv:2409.XXXX (2024).