Extended Abstract

RL for Adaptive Tutoring: When Should a Tutor Intervene?

Peter Alisky, Hoang Nguyen, Zhenghui Chen

Motivation. The role of human tutors isn't just to answer questions; they decide when to help, how much to help, and when to let a student productively struggle. This timing question is central to Bloom's 2-sigma problem [2], which states that individualized instruction can be much more effective than classroom instruction. However, scaling that individualization requires policies that adapt intervention timing to the student state. We study this problem as a partially observable sequential decision problem in which short-term correctness can conflict with long-term learning.

Method. We built a parametric tutoring simulator with a hidden student profile and an 11-dimensional observable state: mastery, recent mistakes, time spent, frustration, confidence, problem difficulty, and recent tutor actions. The tutor chooses among five interventions: direct answer, hint, probing question, easier subproblem, and no intervention. Rewards combine learning gain, engagement, a penalty for unnecessary help, and an end-of-episode retention bonus. We then compare policies with different credit-assignment horizons: supervised one-step oracle imitation, LinUCB contextual bandits, heuristic policies, a recurrent LSTM-DQN agent, and a non-recurrent MLP-DQN ablation.

Experiments. We generated a 75k-transition logged dataset from random, myopic, and strategic rollout policies for supervised learning and evaluation. The main online experiment trains LinUCB, LSTM-DQN, and MLP-DQN directly in the simulator. To test whether the long-horizon result is robust, we ran a 5-seed by 3-setting sweep over the maximum productive-struggle benefit of students, which controls how much delayed reasoning should matter, plus a train $\times$ eval generalization study and an offline-RL (CQL) comparator.

Findings. Both DQN variants lead at every heterogeneity setting in the robustness sweep (5 seeds), with the non-recurrent MLP-DQN strongest throughout: mean episode return rises from 16.1 to 18.5 as productive-struggle benefit increases from 0.02 to 0.20, versus 15.4 to 17.8 for LSTM-DQN. A *balanced* one-step supervised baseline is competitive (15.0 to 17.0), so the RL advantage is real but modest, while offline CQL underperforms every online method (13.2 to 14.5). That the feed-forward MLP-DQN matches or beats the recurrent LSTM-DQN indicates long-horizon Q-learning, not recurrence, is the decisive ingredient in this single-skill simulator.

Interpretation. Behavioral analysis shows that the learned Q policies aren't simply more aggressive. MLP-DQN intervenes on 33.5% of steps and LSTM-DQN on 42.7%, compared with 27.4% for LinUCB, 44.5% for the balanced supervised policy, and 63.2% for the myopic policy. Both DQN variants especially favor easier subproblems, using them on about 32% of steps. These results support the hypothesis that long-horizon RL is useful for adaptive tutoring when productive struggle has delayed value, while also showing that the present state representation already captures much of the information needed for good decisions.

Limitations. Our simulator is synthetic, single-skill, and not calibrated to real tutoring logs. The current work should be viewed as more of a controlled testbed and empirical argument for the importance of horizon-aware intervention policies, not as evidence that the learned policy is ready for deployment with students. Future work should extend the simulator to multi-skill mastery, calibrate it against real educational data, and compare against additional offline and policy-gradient RL methods.

RL for Adaptive Tutoring: When Should a Tutor Intervene?

Peter Alisky
Stanford University
palisky@stanford.edu

Hoang Nguyen
Stanford University
hoanghdn@stanford.edu

Zhenghui Chen
Stanford University
zhengh04@stanford.edu

Abstract

Adaptive tutoring systems must decide not only what information to provide, but when to intervene. Helping too early can reduce productive struggle; helping too late can waste time and increase frustration. We formulate intervention timing as a partially observable Markov decision process and compare policy classes that differ in their credit-assignment horizon. In a controlled tutoring simulator with latent student traits, value-based DQN agents lead a comparison that also includes a balanced supervised one-step oracle, a LinUCB contextual bandit, offline CQL, and heuristic policies. A Modal-parallelized robustness sweep across five random seeds and three productive-struggle settings shows that both DQN variants lead at every heterogeneity level, and that their advantage over the supervised baseline widens when productive struggle has larger delayed benefits, even though the strongest policies reach near-identical final mastery—the gain is efficiency, not raw learning. A properly balanced one-step supervised policy is a competitive baseline, so the RL margin is real but modest. A train $\times$ eval generalization matrix shows the learned policies degrade gracefully under student-model misspecification while the one-step bandit overfits its training population. Two further results sharpen the picture: a non-recurrent MLP-DQN ablation is strongest both in the default environment and across the sweep—so delayed Q-learning, not recurrence, is the central ingredient—and offline CQL, trained on the same logged trajectories under the same reward function used at evaluation, still underperforms every online method, underscoring that online interaction, not merely a long horizon, matters.

1 Introduction

One-on-one tutoring can produce substantially better learning outcomes than standard classroom instruction, a phenomenon often summarized as Bloom's 2-sigma problem [2]. A key part of that advantage is individualization: a human tutor can decide when a student needs direct help, when a hint is enough, and when the student should continue struggling. This project studies one specific form of individualization: intervention timing during problem solving.

Many automated tutoring systems focus on estimating student knowledge or selecting the next problem. Those are important decisions, but within a problem the tutor still faces a sequential control problem. A direct answer may increase immediate correctness while reducing learning; no intervention may preserve productive struggle while increasing frustration. Because these effects unfold over a trajectory, greedy policies can be systematically misaligned with long-term mastery.

We ask whether long-horizon reinforcement learning learns better intervention policies than one-step alternatives. Our hypothesis is that policies with delayed credit assignment will outperform supervised and bandit baselines, especially when students benefit from productive struggle. To test this, we build a simulator in which latent student traits govern learning rate, frustration sensitivity, and struggle benefit. The agent sees only a compact observable state, making the problem partially observable.

2 Related Work

Intelligent tutoring systems have long modeled student knowledge to adapt instruction. Cognitive tutors and model tracing demonstrated that structured models of student understanding can improve pedagogical decisions [1**?**]. Bayesian Knowledge Tracing models latent mastery from response histories [3], and Deep Knowledge Tracing extends this idea with recurrent neural networks [8]. These methods address state estimation, but they do not by themselves specify how much help a tutor should give at each moment.

Reinforcement learning has also been applied to education. Prior work has studied policy evaluation and pedagogical sequencing in educational games [6] and POMDP planning for faster teaching [9]. Our work is closest in spirit to this line: we treat tutoring as sequential decision making. The distinction is that our action space focuses on fine-grained intervention type within a learning episode rather than choosing only the next exercise or lesson.

On the algorithmic side, our Q-learning agents build on DQN [7] with double Q-learning targets [**?**] and, for the recurrent variant, recurrent Q-learning for partially observable environments [4]. We also include an offline comparator, Conservative Q-Learning [**?**], which learns purely from logged trajectories. We compare these agents to LinUCB [5], a contextual bandit that optimizes immediate reward, and to supervised one-step oracle imitation. This comparison isolates the value of long-horizon credit assignment in the tutoring setting, while the MLP-DQN ablation tests whether recurrence is needed given our observable state.

3 Environment and Dataset

State. The simulator exposes an 11-dimensional state: mastery, recent mistakes, time spent, frustration, confidence, problem difficulty, and a 5-step history of tutor actions. Each episode samples a latent student profile with initial mastery, initial frustration, learning rate, frustration sensitivity, productive struggle benefit, and problem difficulty. The policy does not observe these latent traits.

Actions. The tutor chooses among five actions: `direct_answer`, `hint`, `probing_question`, `easier_subproblem`, and `no_intervention`. These actions affect both the student's immediate probability of success and the subsequent mastery/frustration dynamics. For example, direct answers make success likely but produce little mastery gain, while no intervention can yield extra mastery if the student succeeds independently.

Reward. The reward combines learning gain, engagement, and an unnecessary-help penalty at each step. At the end of the episode, a retention bonus rewards high final mastery while penalizing frustration and excessive help. The default horizon is 15 steps, chosen after early experiments showed that longer horizons saturated mastery for most policies.

Logged data. We generated 5,000 episodes (74,970 transitions) using an equal mixture of three rollout policies—random, myopic, and strategic—with 1,666 episodes (24,990 transitions) each, so the logged behavior policy is a deliberately diverse mixture rather than a single near-optimal source. The data is split by episode into an 80/20 train/validation partition (3,998 train / 1,000 validation episodes; 59,970 / 15,000 transitions), ensuring no episode crosses the split. The logged action marginals are 37.5% no intervention, 25.8% hint, 16.0% probing question, 11.5% direct answer, and 9.3% easier subproblem. The supervised baseline trains on a subset of 20,000 transitions relabeled by a Monte Carlo one-step oracle, and offline CQL consumes the logged reward signal directly; online methods, including LinUCB and DQN, train directly against the simulator.

4 Methods

Supervised one-step oracle. For each sampled state, we estimate the action that maximizes immediate simulator reward using a local Monte Carlo oracle. A two-layer MLP classifier with hidden sizes 128 and 64 imitates these oracle labels. This baseline captures what a strong myopic intervention policy can do without delayed credit assignment.

Table 1: Mean episode return across five Modal sweep seeds (seed-to-seed std ≤ 0.5), including MLP-DQN, the balanced supervised baseline, and offline CQL. Both DQN variants lead at every heterogeneity level; CQL trails throughout.

Max struggle benefit	MLP-DQN	LSTM-DQN	Supervised	Bandit	CQL	Myopic	Random
0.02	16.08	15.39	15.03	14.15	13.19	13.20	11.71
0.08	17.55	16.74	15.96	15.82	14.17	14.06	12.00
0.20	18.52	17.84	17.02	16.38	14.45	14.74	12.40

LinUCB contextual bandit. The bandit baseline maintains one linear model per action and chooses the action with the largest upper-confidence-bound score. It observes the same 11-dimensional feature vector as the supervised baseline, but learns online from immediate rewards. It does not bootstrap delayed outcomes.

LSTM-DQN. Our main sweep RL agent is a recurrent DQN. At each step it consumes the current observation, previous action, and previous reward. An LSTM encoder maps this history to a hidden state, and an MLP head predicts Q-values for the five tutor actions. We train with Double DQN targets [**?**], Polyak target-network updates, and epsilon-greedy exploration. The recurrent hidden state is intended to infer the latent student profile from observed responses.

MLP-DQN ablation. To test whether the recurrent state is necessary, we also train a non-recurrent DQN with a two-layer MLP Q-network over only the current 11-dimensional observation. This agent uses the same replay buffer, Double DQN target construction, discount factor, exploration schedule, and training horizon as LSTM-DQN. Because the observable state already includes recent mistakes, confidence, frustration, and a short action history, this ablation tests whether the hand-designed state is sufficient for the current simulator.

CQL (offline RL). To test whether a good tutor can be learned without environment interaction, we train Conservative Q-Learning [**?**] on the same logged dataset used by the supervised and bandit baselines. CQL reuses the LSTM Q-network but adds a conservative regularizer, $\alpha\,\mathbb{E}_s[\log \sum_a \exp Q(s,a) - Q(s,a_{\text{data}})]$, to the TD loss, which suppresses over-optimistic values for actions absent from the data. This isolates the value of *online interaction*: CQL sees the same transitions as the one-step baselines but performs long-horizon credit assignment offline.

5 Experiments

We evaluate policies by rolling them out in the simulator and measuring episode return, final mastery, help fraction, unnecessary-help penalty, and action distribution. The first experiment compares LSTM-DQN, MLP-DQN, LinUCB, supervised imitation, myopic, strategic, and random policies in the default environment. The second experiment uses Modal cloud compute to run a robustness sweep over student heterogeneity for all five learned policies (LSTM-DQN, MLP-DQN, CQL, LinUCB, and supervised imitation) together with the heuristic baselines. We vary the upper bound of productive-struggle benefit over $\{0.02, 0.08, 0.20\}$ and train/evaluate each learned policy with seeds $\{1, 2, 3, 4, 5\}$. The third experiment is a train $\times$ eval *generalization matrix*: we train each learned policy at one productive-struggle distribution T and evaluate it at every distribution $E \in \{0.02, 0.08, 0.20\}$, so the diagonal reproduces the matched sweep and the off-diagonal cells measure performance under student-model misspecification.

All DQN agents train for 3,000 episodes with discount $\gamma = 0.95$, Double DQN targets, Polyak target updates with $\tau = 0.005$, a replay capacity of 5,000 episodes, batch size 64, learning rate 3×10^{-4}, and epsilon-greedy exploration decaying from 1.0 to 0.05 over 1,500 episodes. LinUCB trains for 2,000 episodes with $\alpha = 0.5$. The supervised baseline trains on 20,000 oracle-labeled states with four Monte Carlo samples per local action evaluation, with balanced oversampling of the oracle actions so the classifier can emit any of the five interventions. CQL trains offline for 4,000 gradient steps with $\alpha = 1.0$. The Modal sweep uses 300 rollouts per policy in each seed/setting cell; the default-environment numbers below are the 5-seed means of the struggle $= 0.08$ cell.

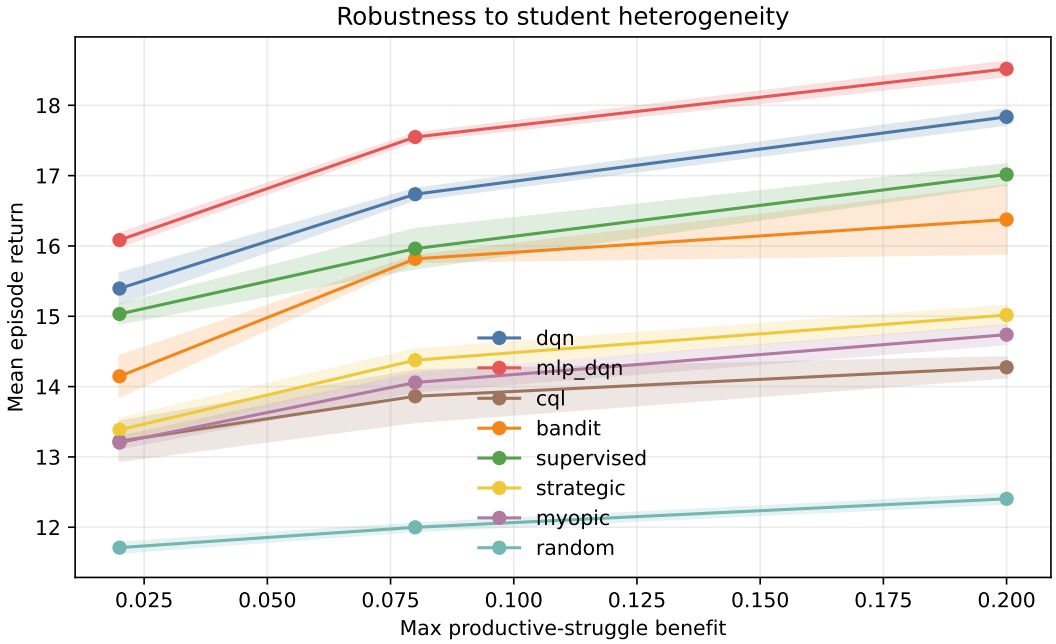

Figure 1: Mean episode return as productive-struggle benefit increases. Bands show 95% confidence intervals across five seeds. Both DQN variants lead at every setting (MLP-DQN best, LSTM-DQN second); the balanced supervised baseline is competitive, and offline CQL trails throughout.

6 Results

The default-environment rollout supports the main hypothesis: both DQN variants outperform the one-step baselines and heuristics. MLP-DQN obtains the highest default return at 17.55, followed by LSTM-DQN at 16.74, the balanced supervised baseline at 15.96, LinUCB at 15.82, strategic at 14.38, offline CQL at 14.17, myopic at 14.06, and random at 12.00. Two points stand out. First, the MLP-DQN result is an important ablation: recurrence is not required in the current single-skill simulator, likely because the observable state already includes a compact action history and student-state summaries. Second, once the supervised classifier is balanced so it can emit every action, it becomes a strong one-step baseline—so the RL advantage, while consistent, is modest rather than dominant.

The robustness sweep strengthens the broader long-horizon RL result. As shown in Figure 1 and Table 1, both DQN variants lead at every heterogeneity level, with MLP-DQN best and LSTM-DQN second, and seed-to-seed std at or below 0.5. The gap between LSTM-DQN and the supervised baseline *widens* with heterogeneity, from 0.36 return at low struggle benefit to 0.82 at high struggle benefit, matching the hypothesis that delayed reasoning matters more as productive struggle becomes more valuable. The advantage is one of *efficiency* rather than raw learning: the strongest policies reach near-identical final mastery (e.g. at the high setting, 0.94 for MLP-DQN versus 0.94 for the bandit and 0.92 for supervised), so the return gap reflects the DQN agents achieving the same learning outcome while spending less unnecessary help. Offline CQL is the exception that proves the rule: it performs the same long-horizon credit assignment as the DQN agents, yet trails every online method at every setting (13.19, 14.17, 14.45). As a control, we generated its logged training rewards under the *same* reward weights used at evaluation, so the gap is not an objective mismatch; its returns are also stable across seeds (std ≤ 0.5), so it is not a tuning artifact. Conservative offline learning from suboptimal-policy logs simply cannot match online interaction here—a direct illustration that online interaction, not merely a long horizon, drives the result.

This pattern matches the motivating hypothesis. When productive struggle has little delayed value, one-step and long-horizon policies are closer. When productive struggle can strongly improve later mastery, the DQN agents benefit from bootstrapping delayed rewards, while the bandit and supervised policies remain tied to immediate rewards or one-step labels.

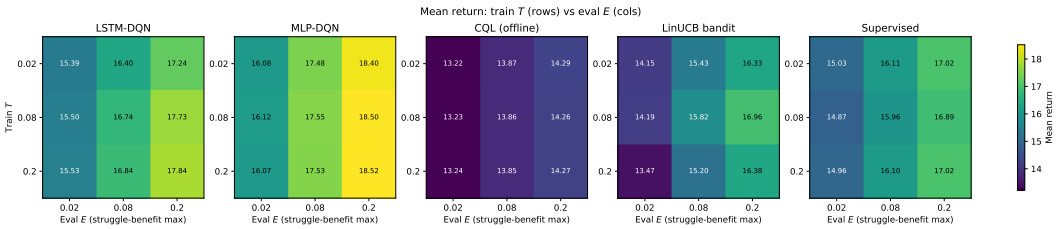

Figure 2: Generalization matrix: mean return for each learned policy, with training distribution T on the rows and evaluation distribution E on the columns. The diagonal is the matched sweep; off-diagonal cells are misspecification. MLP-DQN is nearly invariant to the training distribution, whereas the one-step LinUCB bandit is the most sensitive.

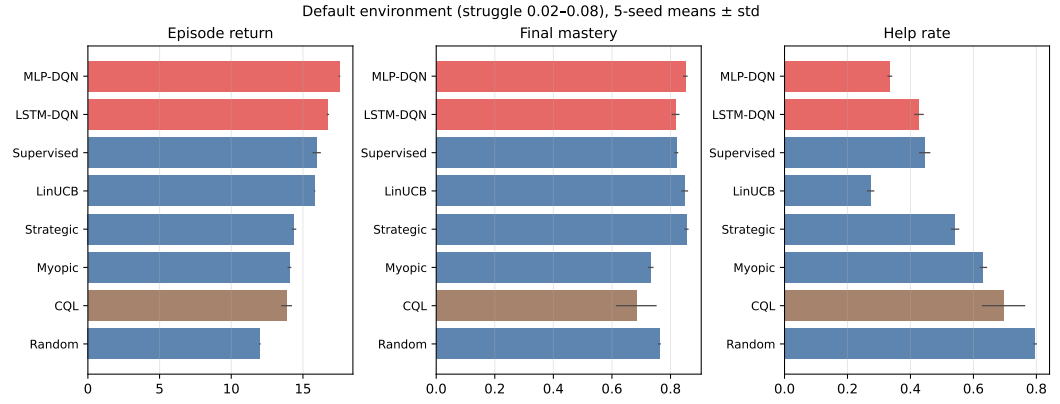

Figure 3: Default-environment rollout performance. MLP-DQN performs best in this setting, while LSTM-DQN still outperforms the one-step baselines. This ablation suggests that long-horizon Q-learning matters more than recurrence for the current hand-designed observable state.

6.1 Generalization under student-model misspecification

A robustness sweep with matched train and evaluation distributions does not reveal what happens when a deployed tutor faces a student population it was not trained on. Our third experiment closes this gap with a train $\times$ eval matrix (Figure 2): each learned policy is trained at one productive-struggle distribution T and evaluated at every E. Because return rises with E for all policies, misspecification is read *within* a column (fixed deployment population, varying the training population).

MLP-DQN is essentially distribution-invariant: its deployment return moves by only about 0.08 regardless of which population it trained on (mean within-column spread 0.08), and the balanced supervised policy is similarly robust (0.15), versus 0.39 for LSTM-DQN and 0.66 for the bandit. The one-step LinUCB bandit is the most brittle, losing about 0.7 return when trained on a high-struggle population and deployed on low-struggle students. Offline CQL sits in between (0.33): more sensitive to its training population than the feed-forward policies, but still less brittle than the bandit, and uniformly weak in absolute terms. No policy collapses, so the misspecification in this simulator is mild, but the ordering is informative: it is the *feed-forward* MLP-DQN, not the recurrent LSTM-DQN, that generalizes best, again indicating that long-horizon credit assignment rather than memory is the decisive factor here.

7 Behavioral Analysis

To understand how the policies differ, we aggregate per-step action choices from the 5-seed sweep rollouts at the default setting. MLP-DQN intervenes on 33.5% of steps and LSTM-DQN on 42.7%, compared with 27.4% for LinUCB, 44.5% for the balanced supervised policy, 63.2% for the myopic policy, and 79.6% for random. Thus the DQN improvement is not explained by simply helping more often: MLP-DQN helps less than the supervised and myopic policies and still achieves higher reward.

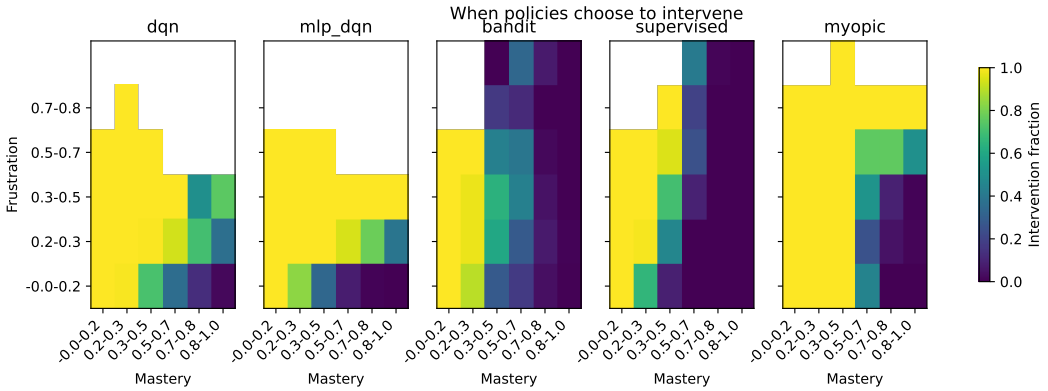

Figure 4: Intervention frequency as a function of mastery and frustration. The DQN variants learn broader state-dependent intervention patterns than the one-step baselines, while the myopic policy over-intervenes in many states.

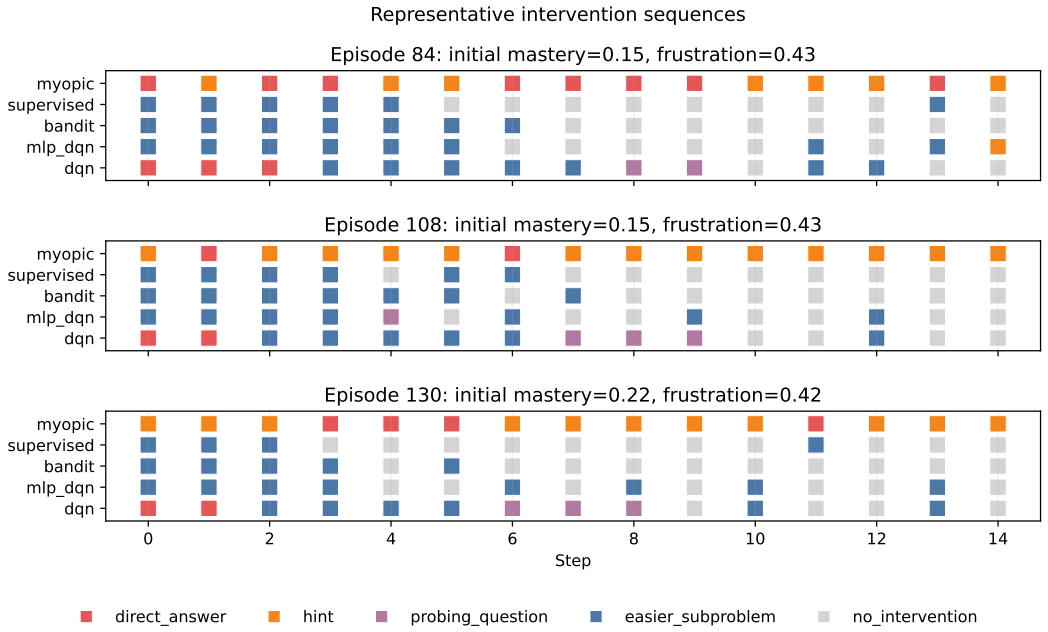

Figure 5: Representative action sequences on matched student profiles. The DQN variants use a mix of no intervention and easier subproblems, whereas supervised and bandit policies more often abstain and the myopic policy intervenes heavily.

The most distinctive behavior is the DQN agents' use of easier subproblems. MLP-DQN selects `easier_subproblem` on 32.3% of steps and LSTM-DQN on 32.2%, compared with 23.9% for LinUCB and 15.5% for supervised imitation. The myopic policy almost never chooses this action (0.8%), preferring direct immediate help. This suggests that Q-learning discovers a useful intermediate strategy: lower the immediate difficulty enough to preserve engagement and learning without giving away the solution.

8 Discussion and Limitations

The main result is that long-horizon Q-learning can learn intervention policies that outperform strong one-step alternatives in a controlled tutoring environment. The heterogeneity sweep is especially important: it shows that the DQN advantage is largest when student productive-struggle effects

are strongest, exactly where immediate-reward optimization should be least adequate. Crucially, this advantage is one of efficiency: the learned policies reach essentially the same final mastery, so the DQN agents win by achieving the same learning outcome with less unnecessary intervention, and by selecting the delayed easier-subproblem strategy that one-step methods never discover. The generalization matrix adds a third lesson: the learned RL policies degrade gracefully under student-model misspecification—MLP-DQN is almost distribution-invariant—whereas the one-step bandit overfits its training population. The MLP-DQN ablation adds a fourth: in this simulator, recurrence is not the decisive factor. The current observable state includes recent actions, mistakes, confidence, and frustration, which appear sufficient for a feed-forward Q-network to perform very well and to generalize best. A harder POMDP, such as a multi-skill environment with longer latent dependencies, may be needed before the recurrent architecture has a clear advantage. A fifth lesson comes from CQL: even when trained on logged data rewarded under the same weights used at evaluation, offline conservative Q-learning underperforms every online method, which separates the contribution of *online interaction* from that of long-horizon credit assignment alone—both appear necessary here.

The project also has clear limitations. First, the simulator is synthetic. Its parameters are plausible but not calibrated against real tutoring logs. Second, the environment tracks a single scalar mastery variable rather than a vector of per-skill mastery. Third, the action space excludes curriculum selection; the tutor controls intervention type but not which skill to practice next. Fourth, while we compare value-based RL (LSTM/MLP-DQN), offline RL (CQL), a contextual bandit, supervised imitation, and heuristics, we do not include a recurrent policy-gradient method such as PPO; that is a natural next comparator. Finally, our recurrent agent uses a simplified DRQN bootstrap in which the target network re-encodes each next-observation sequence from a zero hidden state rather than carrying the rollout's hidden state across the transition. This is a common approximation, but it can only weaken the recurrent target; our finding that the non-recurrent MLP-DQN matches the LSTM is therefore conservative with respect to the LSTM, not an artifact that inflates it. The current study provides a focused empirical test of credit-assignment horizon in tutoring intervention timing.

9 Conclusion

We formulated adaptive tutoring intervention timing as a partially observable reinforcement learning problem and built a simulator for controlled experimentation. Across five seeds, both DQN variants consistently outperformed a balanced supervised one-step baseline, a contextual bandit, offline CQL, and heuristic policies, with the advantage over the supervised baseline widening when productive struggle had larger delayed benefits. The win is modest and one of efficiency—the strongest policies reach near-identical mastery—and the MLP-DQN ablation shows the current state representation makes recurrence unnecessary, so the benefit comes from long-horizon bootstrapping rather than the LSTM itself. That offline CQL underperforms every online method further indicates online interaction is essential, not merely a long horizon. Behavioral analysis shows DQN learns to use easier subproblems as a middle-ground intervention while still allowing students to work independently. These results suggest horizon-aware RL is a promising tool for scalable adaptive tutoring, provided future work validates the environment against real educational data and extends the formulation to multi-skill learning.

10 Team Contributions

- **Peter Alisky:** Led environment design, POMDP formulation, DQN training pipeline, Modal robustness sweep, evaluation infrastructure, and final report integration.
- **Hoang Nguyen:** Developed and analyzed supervised and contextual bandit baselines, supported reward design, and contributed to comparative experiment design.
- **Zhenghui Chen:** Led related-work review, supported student-state modeling and simulator analysis, and contributed to behavioral interpretation and visualization.

Changes from proposal. The division of labor tracked the proposal closely: Peter led the environment and RL training pipeline, Hoang the supervised and bandit baselines and reward design, and Zhenghui the related-work review, simulator analysis, and behavioral visualization. The main role adjustment was that the proposal's planned "robustness" stretch goal grew into a full multi-seed Modal sweep and generalization study, which Peter absorbed into the training-and-evaluation infrastructure;

the offline-RL (CQL) comparator was added during implementation and shared across the team. The proposal emphasized a comparison among supervised, bandit, and RL policies. That core plan remained intact. We added a stronger POMDP formulation by hiding latent student traits, retuned the horizon and reward to avoid mastery saturation, and added a Modal robustness sweep (five seeds) over productive-struggle heterogeneity, a train $\times$ eval generalization study, and an offline-RL comparator (CQL). We also fixed a degenerate supervised baseline by balancing its training labels, which made it a fair and competitive one-step comparator. We did not complete the multi-skill extension or a recurrent policy-gradient (PPO) comparator; instead, we prioritized reproducible experiments, statistical robustness, and behavioral analysis for the core hypothesis.

11 AI Tools Disclosure

The team used AI coding assistants, primarily Claude (Anthropic's Claude Code), to help with boilerplate, debugging, experiment orchestration, code auditing, documentation, and drafting support. Our core research decisions, including our primary algorithms, the experimental design, interpretation of results, and final responsibility for the code and report remained with the team. We used AI as an aid for our implementation, but not as substitue for our understanding of methods and results.

References

[1] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[2] Benjamin S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[3] Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 1994.

[4] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[5] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.

[6] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 1077–1084, 2014.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[8] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, 2015.

[9] Anna N. Rafferty, Emma Brunskill, Thomas L. Griffiths, and Patrick Shafto. Faster teaching by pomdp planning. In *Artificial Intelligence in Education*, pages 280–290, 2016.

A Additional Implementation Details

The codebase includes scripts for dataset generation, supervised training, LinUCB training, DQN training, evaluation, plotting, Modal sweeps, and behavioral analysis. The main reproducibility commands are documented in the repository's `docs/FINAL_SUBMISSION_CHECKLIST.md`.