Noname manuscript No.
(will be inserted by the editor)

The Impact of Batch Deep Reinforcement Learning on Student Performance: A Simple Act of Explanation Can Go A Long Way

Markel Sanz Ausin · Mehak Maniktala · Tiffany Barnes · Min Chi

Received: date / Accepted: date

Abstract While Reinforcement learning (RL), especially Deep RL (DRL), has shown outstanding performance in video games, little evidence has shown that DRL can be successfully applied to human-centric tasks where the ultimate RL goal is to make the *human-agent interactions* productive and fruitful. In real-life, complex, human-centric tasks, such as education and healthcare, data can be noisy and limited. Batch RL is designed for handling such situations where data is *limited yet noisy*, and where *building simulations is challenging*. In two consecutive empirical studies, we investigated Batch DRL for pedagogical policy induction, to choose student learning activities in an Intelligent Tutoring System. In Fall 2018 (F18), we compared the Batch DRL policy to an Expert policy, but found no significant difference between the DRL and Expert policies. In Spring 2019 (S19), we augmented the Batch DRL-induced policy with *a simple act of explanation* by showing a message such as *"The AI agent thinks you should view this problem as a Worked Example to learn how some new rules work."*. We compared this policy against two conditions, the Expert policy, and a student decision making policy. Our results show that 1) the Batch DRL policy with explanations significantly improved student learning performance more than the Expert policy; and 2) no significant differences were found between the Expert policy and student decision making. Overall, our results suggest that *pairing simple explanations with the Batch DRL policy* can be an important and effective technique for applying RL to real-life, human-centric tasks.

Keywords Deep Reinforcement Learning · Pedagogical Policy · Explanation

Markel Sanz Ausin
North Carolina State University
E-mail: msanzau@ncsu.edu

Min Chi
North Carolina State University
E-mail: mchi@ncsu.edu

1 Introduction

Reinforcement Learning (RL) is one of the most promising approaches to induce effective policies that determine the best *action* for *an agent* to take in any given *situation*, so as to maximize a cumulative *reward*. In recent years, deep neural networks have enabled significant progress in RL research and Deep Reinforcement Learning (DRL) has been shown to be a very powerful technique on a wide range of applications. For example, Deep Q-Networks (DQNs) [43] have successfully learned to play Atari games at or exceeding human level performance by combining deep convolutional neural networks and Q-learning. Since then, DRL has achieved notable successes in a variety of complex tasks such as robotics control [2] and the game of Go [62]. From DQN, various DRL methods such as Double DQN [66] or Actor-Critic methods [53,54] were proposed and shown to be more effective than the classic DQN. Despite DRL's great success, there are still many challenges preventing DRL from being applied more broadly in practice, including applying it to educational systems. One major problem is sample inefficiency of current DRL algorithms. For example, it takes DQN hundreds of millions of interactions with the environment to learn a good policy and generalize to unseen states, while we seek to learn policies from datasets with around 800 student-tutor interaction logs or fewer and they are often noisy, given the nature of the task. In this work, we used *batch RL*, which is a sub-field of RL that deals with the inability to explore the environment, and all the learning is induced from a *fixed*, pre-existing limited yet noisy dataset that was obtained from human-agent interactive environments using some unknown behavior policy.

In our work, we applied Batch DRL to induce pedagogical policies that specify how each problem should be presented to the students. In our approach, we employ a Gaussian Process (GP) method for automatically inferring the immediate rewards from delayed ones in our training corpus and then use those results to train a DQN agent for making pedagogical decisions. In Fall 2018, we empirically evaluated the batch DRL policy against an expert-designed policy and found no significant differences (as described in Section 6). Such results, while disappointing, were in line with a large body of research on improving ITS effectiveness by applying RL (e.g. [18,13,50,60,3,57,12]). So the question is: *can pedagogical policies induced by our batch DRL be more effective than the Expert designed rules?* In this work, we investigated two aspects related to pedagogical decisions: *communication* and *agency*.

Communication: while RL especially DRL has achieved superhuman performance in several complex games [62,63,68,2] where the ultimate goal is to make the agent effective, in human-centric tasks such as ITSs, the ultimate goal is for the agent to make the *human-system interactions* productive and fruitful. Therefore, we argue it is important to *communicate* the agent's pedagogical decisions to students. Prior work on applying RL to ITSs primarily focused on inducing effective pedagogical policies *for the tutor to act*, but the tutor rarely "explains" to students *why* certain pedagogical decisions are made. As far as we know, no prior research has been done on exploring the effective-

ness of explaining pedagogical policies to students. On the other hand, prior research in Self-Determination Theory (SDT) suggests that explanations could be a powerful tool to increase student engagement and autonomy in learning. For example, it was shown that explaining the benefits of learning a specific task to students would increase their sense of control over their own learning, which can improve their learning outcomes [15,52,31,29,70,61]. In this work, we address the limitation of communication by providing students with simple explanations of the Batch DRL pedagogical policy. We hypothesize that combining a DQN-based policy with simple explanations would outperform an Expert policy in that the former would result in better student learning performance **(Communication Hypothesis)**.

Agency: who should be the decision maker, the student or the tutor? Rather than inducing pedagogical policies so that the tutor can decide effectively, would it be more effective if we just let students make certain pedagogical decisions? Prior research has shown that it is desirable for students to experience a sense of control over their own learning, which could enhance their motivation and engagement [15,29,1] and improve their learning experience [52,70]. People are more likely to persist in constructive activities, such as learning, exercising, or quitting smoking, when they are given choices and make decisions.Thus, we investigated the effectiveness of letting students make pedagogical decisions vs. the traditional tutor-driven Expert rule approach. We hypothesize that letting students make their own pedagogical decisions can be more effective than an Expert policy **(Agency Hypothesis)**.

Through two empirical classrooms studies, our results show that in terms of the Communication Hypothesis, the Batch DRL policy with explanations can improve student learning performance more than our Expert policy (Spring 2019) while the Batch DRL policy without explanations does not; in terms of Agency Hypothesis, no significant difference was found between student decision making and the Expert policy. In summary, this work suggests that neither letting the tutor make Expert or DRL-induced pedagogical policy decisions alone, nor letting students make decisions alone, may be sufficient to improve student learning. A more effective way may be letting the tutor make effective pedagogical decisions while communicating the decisions with students through simple explanations.

2 Background & Related Work

2.1 Deep Reinforcement Learning (DRL) and Batch DRL

In recent years, the combination of deep learning with neural networks and novel RL algorithms has made solving complex problems with DRL possible. As an example of DRL, the Deep Q-Network (DQN) algorithm [43] takes advantage of convolutional neural networks to learn to play Atari games by observing the pixels directly. Since then, DRL has achieved success in various complex tasks, such as robotic control [2] and playing games, including Go

[62], Chess/Shogi [63], and Starcraft II [68]. One major challenge of these methods is *sample inefficiency* where RL policies need large sample sizes to learn optimal, generalizable policies.

Batch RL [33], a sub-field of RL, aims to fix this sample inefficiency problem by learning the optimal policy from a fixed set of a priori-known transition samples, efficiently learning from a potentially small amount of data, to generalize to unseen parts of the environment. Batch RL has also been investigated in combination with Neural Networks, resulting in Batch Deep Reinforcement Learning or Batch DRL. Several different Batch DRL algorithms have been developed [16,32,34], and some of them are more effective than others at solving offline RL tasks [20]. In the work by Fujimoto et al. [21], they added a constraint to the optimization problem, restricting the action space to make the agent behave similarly to the training data. Another similar approach is to pre-train a model on the training data and use it as a strong prior, and combine it with KL-control to penalize divergence from this prior during RL training, as shown by Jaques et al. [26]. However, all of these methods have been evaluated on agent-centric tasks like game playing, with no human involved, whereas our goal is to apply Batch DRL to learn a pedagogical policy for an ITS. It is not clear whether these DRL methods will be effective on such a task.

2.2 RL for Pedagogical Policy Induction

Prior research in applying Reinforcement Learning (RL) to pedagogical policy induction can be roughly divided into classic RL vs. DRL approaches. Generally speaking, there are two major categories of RL: online and offline/batch. Online RL algorithms learn a policy while the agent interacts with the environment; Offline/Batch RL algorithms learn the policy from pre-collected training data. More specifically, they "take advantage of previous collected samples, and generally provide robust convergence guarantees" [55]. Online RL methods are generally appropriate for domains where the state representation is clear and interacting with simulations and actual environments is relatively computationally cheap and feasible, so most prior work on DRL mainly took an online learning approach. On the other hand, for domains such as education, building accurate simulations or simulating students can be especially challenging because human learning is a rather complex, not fully understood process; moreover, learning policies while interacting with students may not be feasible and more importantly, may not be ethical. Therefore, some of prior work on applying RL to ITSs is offline/batch. This approach was achieved by, first, collecting a training corpus. One common convention, and the one used in our study, is to collect an *exploratory* corpus by training a group of students on an ITS that makes random yet *reasonable* decisions and then apply RL to induce pedagogical policies from that exploratory training corpus. An empirical study was then conducted from a new group of human subjects interacting with different versions of the system. The only difference among

the versions was the policy employed by the ITS. Lastly, the students' performance was statistically compared. Due to cost limitations, typically, only the *best* RL-induced policy was deployed and compared against some baseline policies.

Prior research using classic RL approaches has applied both online and offline/batch approaches to induce pedagogical policies for ITSs [18]. For online RL, Beck et al. [10] applied temporal difference learning to induce pedagogical policies to minimize student time on task. Similarly, Iglesias et al. applied Q-learning to induce policies optimized for efficient student learning [24,25]. More recently, Rafferty et al. applied an online partially observable Markov decision process (POMDP) to induce policies for faster student learning [47]. All of these models were evaluated via simulations or classroom studies, yielding improved student learning and/or behaviors as compared to some baseline policies. For offline RL, Shen et al. applied value iteration and least squares policy iteration on a pre-collected exploratory corpus to induce a pedagogical policy that improved student learning [59,58]. Chi et al. applied policy iteration to induce a pedagogical policy aimed at improving student learning gains [14]. Mandel et al. [35] applied an offline POMDP to induce a policy which aims to improve student performance in an educational game. Zhou et al. induced a Gaussian Processes-based hierarchical RL policy to improve student learning gains [71]. All of these Batch RL models were evaluated in classroom studies, yielding improved student learning or performance relative to a baseline policy.

More recently, some researchers applied both online and offline/batch DRL approaches to induce pedagogical policies for ITSs as well [5,27,72]. For example, we have applied batch DRL for inducing a pedagogical policy on an ITS, and showed that the induced policy is indeed effective in improving learning of students with high initial competency, but not the low ones [3]. Wang et al. applied an online DRL approach to induce a policy for adaptive narrative generation in educational game using simulations [69]; the resulting DRL-induced policies were evaluated via simulations only. In this work, based on the characteristics of our task domain, we focus on batch RL with neural networks, also known as batch Deep Reinforcement Learning (batch DRL) [26,20] and evaluate their effectiveness in classroom studies.

3 Methods

In conventional RL, an agent interacts with an environment $\mathcal{E}$ over a series of decision-making steps, which can be framed as a Markov Decision Process (MDP). At each timestep t, the agent observes the environment $\mathcal{E}$ in state s_t; it chooses an action a_t from a discrete set of possible actions; and $\mathcal{E}$ provides a scalar reward r_t and evolves into the next state s_{t+1}. The future rewards are discounted by the factor $\gamma \in (0, 1]$. The return at time-step t is defined as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the last time-step in the episode. The agent's goal is to maximize the expected discounted sum

of future rewards, also known as the return, which is equivalent to finding the optimal action-value function $Q^*(s,a)$ for all states. Formally, $Q^*(s,a)$ is defined as the highest possible expected return starting from state s, taking action a, and following the optimal policy π^* thereafter. It can be calculated as $Q^*(s,a) = \max_\pi \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$ and $Q^*(s,a)$ must follow the *Bellman Equation*. We follow the batch RL formulation in that we have a fixed-size dataset $\mathcal{D}$ consisting of all historical sample episodes, each formed by a sequence of state, action, reward tuples $(s_0, a_0, r_0, ..., s_T, a_T, r_T)$. We assume that the state distribution and behavior policy that were used to collect $\mathcal{D}$ are both unknown. We explored two batch DRL algorithms: *Deep Q-Network* (DQN) and *Double Deep Q-Network* (Double DQN).

3.1 DQN

The DQN algorithm [43] is fundamentally a version of Q-learning which uses neural networks to approximate the true Q-values. In order to train the DQN algorithm, two neural networks with equal architectures are employed: one for calculating the Q-value of the current state and action $Q(s,a)$, and another neural network to calculate the Q-value of the next state and action $Q(s',a')$. The former is the main network and its weights are denoted $\boldsymbol{\theta}$ and the latter is the target network, and its weights are denoted $\boldsymbol{\theta}^-$. Equation 1 shows its corresponding *Bellman Equation*. It is trained through gradient descent to minimize the squared difference of the two sides of the equality. Online DQN uses an experience replay buffer to store the recently collected data and to uniformly sample (s,a,r,s') steps from it. When inducing our batch RL policy, the whole $\mathcal{D}$ is in the experience replay buffer.

$$Q(s,a;\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{s'\sim\mathcal{E}}[r + \gamma \max_{a'} Q(s',a';\boldsymbol{\theta}^-)] \tag{1}$$

The main network is trained on every training iteration, while the target network is frozen for a number of training iterations. Every n training iterations, the weights of the main neural network are copied into the target network. This is one of the techniques used in order to avoid divergence during the training process. In practice, DQN also uses an experience replay buffer to store the recently collected data and to uniformly sample (s,a,r,s') steps from it. By sampling uniformly, it breaks the correlations between samples of the same episode, making the learning process more robust and stable. In this work, as we are doing batch RL, our whole dataset will be the experience replay buffer, and it will not change during the training process.

3.2 Double-DQN

To improve upon the weaknesses of DQN, the Double-DQN algorithm was proposed by Van Hasselt et al. [66] by combining the idea behind Double Q-Learning [23] with the neural network advances of the DQN algorithm to form

Double-DQN. The intuition behind it is to decouple the action selection from the action evaluation. To achieve this, the Double-DQN algorithm uses the main neural network for action selection first, and then the target network evaluates its Q-value. This trick has been proven to significantly reduce overestimation in Q-value calculations, resulting in better final policies. With this technique, the modified *Bellman Equation* becomes:

$$Q(s,a;\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{s'\sim\mathcal{E}}[r + \gamma Q(s', \operatorname*{argmax}_{a'} Q(s',a',\boldsymbol{\theta});\boldsymbol{\theta}^{-})] \tag{2}$$

3.3 Inferring the Immediate Rewards

One of the most important problems that needs to be solved to effectively induce an effective Batch DRL policy is the Credit Assignment Problem [42]. When the rewards have a temporal delay, it can be challenging to appropriately assign reward/blame for each of the actions in a sequence, and Reinforcement Learning algorithms struggle to learn a correct value or policy function for each state and action pair. This problem is particularly relevant when all the learning has to be performed offline, as in the case of Batch DRL, and when there is limited data available. Solving the Credit Assignment Problem would allow us to induce more effective DRL policies for education, and less data would be required, which would ultimately result in improved student learning. For this reason, to tackle the delayed reward problem, in this work we follow the Gaussian Processes (GP) approach described in [7,4,6] to estimate immediate rewards given the delayed rewards for a historical dataset $\mathcal{D}$ consisting of m trajectories, and containing n unknown immediate rewards. The delayed rewards are used to calculate the immediate rewards, employing a minimum mean square error (MMSE) estimator in the Bayesian setting [28, 19, 22]. Assume that $\mathbf{R} = \mathbf{D}\mathbf{r} + \varepsilon$ is a linear process where $\mathbf{D}$ is a known linear model matrix that is used to associate the immediate rewards with the delayed rewards in the same episode. In that equation, $\mathbf{r}$ is a $\mathbb{R}^{n\times 1}$ random vector of unknown immediate rewards, $\mathbf{R}$ is a $\mathbb{R}^{m\times 1}$ vector of observed delayed rewards and ε is a vector of independent and identically distributed noise with mean of zero and standard deviation of $\sigma_{\mathbf{R}}$. In order to train the GP, we assume that the discounted sum of the immediate rewards is equal to the delayed rewards for each episode.

We assume that the immediate rewards follow a Gaussian Process defined as $\mathbf{r} \sim \mathcal{N}(\mu_{\mathbf{r}}, \mathbf{C_{rr}})$ where $\mu_{\mathbf{r}}$ is the a priori mean and $\mathbf{C_{rr}}$ is the a priori covariance defined by an appropriate kernel [8]. Using the theorem of conditional distribution of multivariate Gaussian distributions [48], we can calculate the conditional expectation of the immediate rewards given the delayed rewards as:

$$\mathbb{E}[\mathbf{r}|\mathbf{R}] = \mu_{\mathbf{r}} + \mathbf{C_{rr}}\mathbf{D}^{T}{\mathbf{C_{RR}}}^{-1}(\mathbf{R} - \mathbf{D}\,\mu_{\mathbf{r}}) \tag{3}$$

and we can calculate the posterior covariance of the inferred immediate rewards given the delayed rewards as:

$$\mathbb{C}[\mathbf{r}|\mathbf{R}] = \mathbf{C_{rr}} - \mathbf{C_{rr}}\mathbf{D}^T{\mathbf{C_{RR}}}^{-1}\mathbf{D}\mathbf{C}_{\mathbf{rr}}^T \tag{4}$$

where $\mathbf{C_{RR}} = \mathbf{D}\mathbf{C_{rr}}\mathbf{D}^T + \sigma_{\mathbf{R}}^2\mathbf{I}$ and $\mathbf{I}$ is the identity matrix.

Algorithm 1 shows how we inferred the immediate rewards. Estimation of the mean and covariance of the random column vector $\mathbf{r}$ in Equations 3 and 4 requires the inverse of the matrix $\mathbf{C_{RR}}$. By introducing several intermediary variables, this algorithm provides an efficient solution to matrix inversion using the Cholesky decomposition similar to the Gaussian Processes algorithm implementation [48].

Algorithm 1 Immediate reward approximation algorithm.

Inputs: $\mathbf{R}$, $\mu_{\mathbf{r}}$, $\mathbf{C_{rr}}$, $\mathbf{D}$, $\sigma_{\mathbf{R}}^2$
$\mathcal{L} = \text{Cholesky}\left(\mathbf{D}\mathbf{C_{rr}}\mathbf{D}^{\mathrm{T}} + \sigma_{\mathbf{R}}^2\mathbf{I}\right)$
$\boldsymbol{\beta} = \mathcal{L}\backslash(\mathbf{R} - \mathbf{D}\,\mu_{\mathbf{r}})$ forward-substitution algorithm
$\boldsymbol{\alpha} = \mathcal{L}^{\mathrm{T}}\backslash\boldsymbol{\beta}$ back-substitution algorithm
$\overline{\boldsymbol{k}} = \mathbf{D}\mathbf{C}_{\mathbf{rr}}^{\mathrm{T}}$
$\mathbf{v} = \mathcal{L}\backslash\overline{\boldsymbol{k}}$
$\mathbb{E}[\mathbf{r}|\mathbf{R}] = \mu_{\mathbf{r}} + \overline{\boldsymbol{k}}^{\mathrm{T}}\boldsymbol{\alpha}$
$\mathbb{C}[\mathbf{r}|\mathbf{R}] = \mathbf{C_{rr}} - \mathbf{v}^{\mathrm{T}}\mathbf{v}$
return: $\mathbb{E}[\mathbf{r}|\mathbf{R}]$ and $\mathbb{C}[\mathbf{r}|\mathbf{R}]$

4 Our Logic Tutor and Context

Deep Thought (DT) is a data-driven Intelligent Tutoring System (ITS) for open-ended multi-step propositional logic problems with data-driven features including next-step hints [64,9], adaptive assistance [37], and pedagogical policies for worked example presentation induced via reinforcement learning [59, 60,11,3]. In this section we describe the tutor, its population, and the curricular context.

4.1 DT Tutor Interface

Figure 1 shows the tutor interface: the left window is the *workspace* where students construct solutions, the central window lists the domain rule buttons, and the right window provides instructions and information such as the rules that are meant to be practiced in the current problem [36]. The pedagogical policy decides whether to represent each problem in the training levels 2-6 as a Worked Examples (WE) or as a Problem Solving (PS). Figure 1 shows the user interface for PS, and Figure 2 shows the interface for WE. Each logic statement is graphically represented as a *node*. Deep Thought shows several problem-provided statements (that are meant to be used as existing or known facts) at the top of the workspace, and a conclusion to derive at the

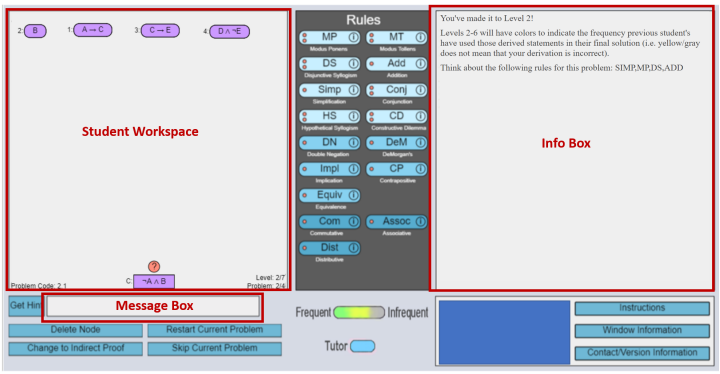

Fig. 1: Problem Solving (PS) Interface in Deep Thought

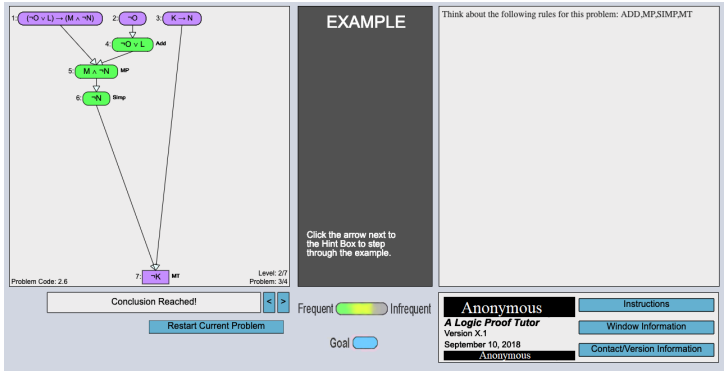

Fig. 2: Worked Example (WE) Interface in Deep Thought

bottom. In a PS problem, students carry out problem-solving steps by deriving new statements from old ones using domain rules. This is a typical procedure used across STEM domains to apply principles or rules to known information to derive new facts [45]. In this work, a problem-solving step consists of a new *derived statement* and its *justification*, where the justification includes specifying the domain rule and the source statements used to show the new derived statement is true. Problem-solving continues until the conclusion is the derived statement in a step that is justified.

Figure 1 shows an example problem with three nodes 1-4 for the problem-provided statements (2: B, 1: $A \rightarrow C$, 3: $C \rightarrow E$, 4: $D \wedge \neg E$) at the top of the workspace. The conclusion to be derived (C: $\neg A \wedge B$) is at the bottom, with a question mark above it indicating that it is not yet justified. Each problem-solving step involves the same process: clicking on 1-2 source nodes and a rule button, and entering the new derived statement. The tutor verifies whether the source nodes and rule correctly justify the derived statement. Once verified, a new node appears, colored based on the frequency of its necessity to previous

student solutions[1], where green is frequent, yellow is infrequent, and gray is never. These colors provide an indication of whether a step is in an optimal problem-solving path.

4.2 Problem Solving in DT

We now walk through a student's experience of solving the problem in Figure 1 to obtain the solution shown in Figure 3 [36]. First, the student clicks on node 4 and rule Simp, and types the new derived statement, D. The tutor verifies the correct justification, and draws node 5, labeled with Simp and an arrow from node 4 to 5. Next, the student applies the same process to derive and justify node 6, which is green since it was frequently necessary in historical solutions. To derive node 7, the student clicks node 1 and rule Impl, and types in the derived statement $\neg A \vee C$. The student then clicks "Get Hint" to request a hint, and "Try to derive $\neg C$" appears in the message box. Additionally, when a hint is shown, the Info Box (Figure 1) also shows the information of the node that should be derived next, but no help is given about what rules should be used to derive that node. Next, the student tries to follow the hint by selecting nodes 3 and 6 and the rule MP. The tutor detects this incorrect rule application, records the error in the data log, and provides a text-based descriprion of what caused the failure during the rule application, but since it was a mistake, no new node is created. Since nodes 3 and 6 are still selected, the student clicks on the correct rule – MT, and types in the derived statement $\neg C$. This process correctly *justified* the hint content statement $\neg C$, so node 8 appears with MT with arrows from nodes 3 and 6. The student similarly clicks on nodes 7 and 8, and rule DS to derive node 9. Finally, the student clicks on nodes 2 and 9, and rule Conj to derive the conclusion, and the tutor detects that the problem is complete.

On the other hand, in a worked example, the tutor shows the most efficient solution to logic problems. Students can navigate back and forth through the problem-solving steps carried out by the tutor using the left/right arrows shown in Figure 2, but are not asked to justify these steps. The Message Box also shows a short description of the last node derivation shown in each step.

4.3 Tutor Pretest, Training, and Posttest

The tutor provides students with practice solving logic problems, divided into four sections: introduction, pretest, training, and posttest. The introduction presents two worked examples to familiarize students with the tutor interface. Next, students solve two problems in a *pretest*. Pretest problems are straightforward, with short optimal solution lengths ($Mean = 3.5$, $SD = 0.71$). Next, students are assigned to *training* conditions using stratified sampling on the

[1] A node is 'needed' when its deletion would make a solution incomplete.

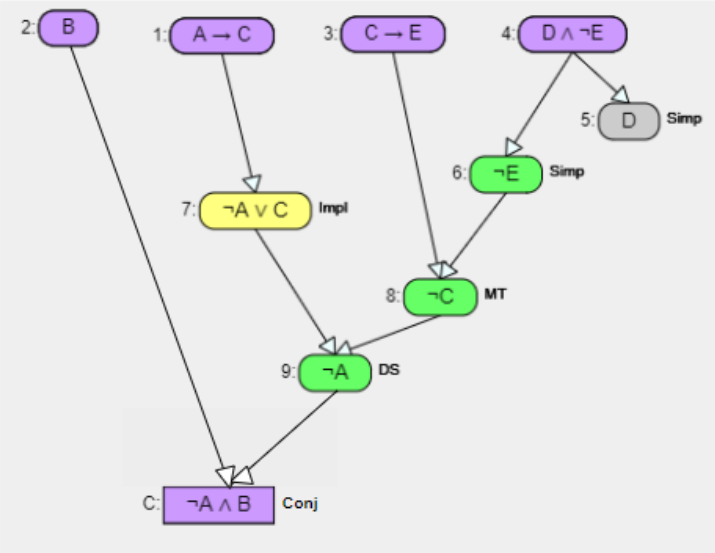

Fig. 3: A walkthrough of problem solving in Deep Thought.

pretest performance, where they complete 20 problems spanning five training levels. Each level contains a set of eight problems, which consist of seven training problems (ordered linearly in difficulty) and a level posttest (most difficult problem in the set). Upon starting a new level, students receive a problem (PS or WE) with medium difficulty (no. #4 in the linear ordering of difficulty), and then the problem selector selects the next problem adapting to student performance in the current problem. Students may skip up to three PS problems per level, with each skip taking them to an easier PS problem. A skipped problem is considered an incomplete solution. Students may also restart problems. For each level, students must complete *three* training problems (WE, or PS with hints), after which they must solve the level posttest problem (PS with no hints). Training problems are of medium difficulty, with optimal solution lengths: $Mean = 4.99$ steps, ($SD = 1.32$). Training conditions differ only in the pedagogical policy used to select which level training problems are WE or PS. Finally, after the training phase is complete, students take a more difficult *posttest* with four problems, with longer optimal solution lengths compared to the other sections ($Mean = 7.25$, $SD = 1.89$).Students always receive immediate feedback on rule application errors for all the PS problems in the pretest, training phase, and posttest. However, hints are only available in the training phase.

The score used to evaluate the problem-solving ability of students in the pretest and posttest problems is calculated using two key metrics: the number of incorrect actions taken by the student, and the time spent in the problem. Making fewer mistakes in a problem and solving it faster will result in higher scores. However, the grade obtained by the students is not affected by this

scoring system, it is for research use only and not shared with the students. This score function was defined by the professor who teaches the course, who has over 20 years of experience on the subject.

4.4 Population and Curricular Context

DT is used in the undergraduate Discrete Mathematics course at North Carolina State University. The course is typically composed of about 60% sophomores, 30% juniors, 9% seniors, and 1% freshmen. The course is taken by approximately 32% of the students in the NC State College of Engineering, including computer science, computer engineering, and electrical engineering majors. In Fall 2018, college demographics include 25.3% women, 67.2% white, 8.3% Asian, 6.5% Non-resident Alien, 0.3% American Indian/Native American, 3.3% Black/African American, 4.8% Hispanic/Latinx, 4.83% from two or more underrepresented minorities, and 5.9% with unknown race/ethnicity[2].

The empirical studies were carried out in the Fall 2018 (F18) and Spring 2019 (S19) semesters, where our ITS was given as one of the regular homework assignments, and students had one week to complete it.

5 Pedagogical Decisions & Pedagogical Policy Induction

The tutor includes several training conditions with different pedagogical policies, with students randomly assigned using stratified sampling on the pretest to balance incoming competence across conditions. This allows us to perform empirical classroom studies to compare pedagogical policies. Our baseline pedagogical policy is designed by the instructor, who has more than 20 years of experience on the subject, and it is referred to as the *Expert* policy. Based on our ITS, prior instructional experience, and prior research on WE vs. PS, the *Expert* policy consists of alternating between PS and WE, so if a problem is shown as a PS the next one will be a WE. The exception for this rule is the last problem in each level, which is fixed as a PS for all the policies.

5.1 Pedagogical Decisions

When comparing the effectiveness of students' pedagogical decision-making vs. batch DRL, we control the instructional content to be equivalent for all students in that our ITS *uses the same problem selection algorithm for all students*, and we focused on tutorial decisions that cover the same domain content: Problem-Solving (PS) versus Worked Examples (WE). As described above, in PS, students are given tasks or problems to complete either independently or with assistance of ITSs while in WE, students are given detailed solutions.

[2] More details can be found on Fall 2018 student demographics at NCSU at https://www.engr.ncsu.edu/ir/fast-facts/fall-2018-fast-facts/

A great deal of research has investigated the impacts of WEs vs. PSs on learning. [65,41,39,38,49,56,44,51]. Generally speaking, it is shown that studying WEs can significantly reduce the total time on task while keeping the learning performance comparable to doing PS [41,39,38]; alternating WE and PS can be *more effective* than PS only [65,38,49,56,44,51]. Despite prior work, there is little consensus on how they should be combined effectively and thus when deciding between PS and WE, most existing ITSs always choose PS [30,67]. Since there is no widespread consensus on how or when each alternative should be used, we apply batch DRL to derive pedagogical strategies directly from empirical data.

5.2 Training Corpus

Our training corpus consists of 786 historical student-ITS trajectory interactions over 5 different semesters, one trajectory per student. All students go through a standard pretest, training on ITS, and posttest procedure and each student spent around 2-3 hours on the ITS. To represent the learning environment, 142 state features from five categories were extracted:

- **Autonomy:** 10 features describing the amount of work done by the student. This category describes the amount of work the student has done recently or in a certain period of time.
- **Temporal:** 29 time-related information features about the student's behavior, such as the average time per step *avgStepTime*, or the total time on training so far *timeOnTutoring*. It also includes the time spent on PS, the time spent on WE, and so on. This category reflects the student's working speed or the amount of effort he/she has put into learning.
- **Problem Solving:** 35 features such as the difficulty of the current problem, the number of easy and difficult problems solved on the current level, the number of PS and WE problems seen in the current level, or the number of nodes the student added in order to reach the final solution. This category also provides information about the current task the student is working on.
- **Performance:** 57 features such as the number of incorrect steps, and the ratio of correct to incorrect rule applications for different types of rules. This category reflects the student's current competence level.
- **Hints:** 11 features such as the total number of hints requested or the number of hints the tutor provided without the student asking for them. This category describes the student's hint usage behavior.

The primary goal of our RL-induced pedagogical policy is to improve student Learning Gain, measured by the difference between the posttest and the pretest scores with a range of $[-200, +200]$. Since in RL immediate rewards are often more efficient than delayed ones, here we applied Gaussian Processes (GP) [7] to infer the immediate rewards for non-terminal states from the final delayed reward (students' Learning Gain).

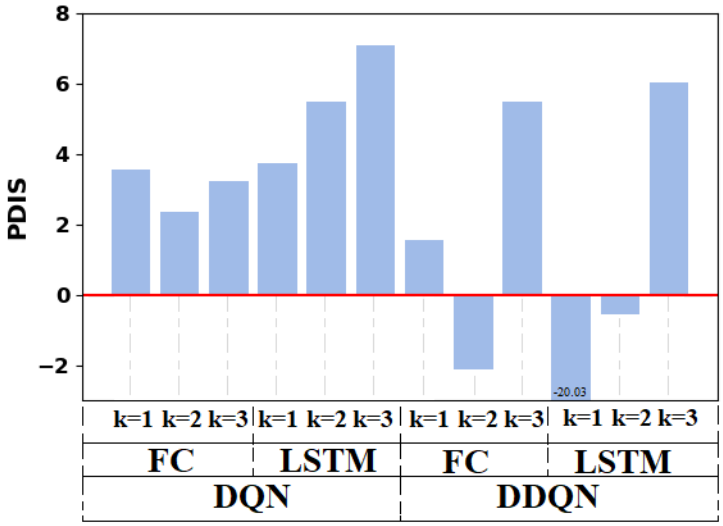

Fig. 4: Importance sampling results. The x axis shows the twelve candidate neural network models and the y axis shows the PDIS values for each model (the higher the better). The red line shows the PDIS value for a random target policy.

5.3 Policy Induction

For both DQN and Double DQN, we explored using Fully Connected (FC) vs. Long Short Term Memory (LSTM) to estimate the action-value function Q. Our FC network consists of four fully connected layers of 128 units each, with Rectified Linear Unit (ReLU) as the activation function. Our LSTM architecture consists of two layers of 100 LSTM units each with ReLU activation functions, and a fully connected layer as output. Additionally, for both FC and LSTM, for a given time t, we explored three input settings: 1) $k = 1$ that use only the last state s_t; 2) $k = 2$ that uses to use the last two states: s_{t-1} and s_t; and 3) $k = 3$ for using s_{t-2}, s_{t-1} and s_t. L2 regularization was employed to get a model that generalizes better to unseen data and avoid overfitting. We trained our models for 50,000 iterations, using a batch size of 200.

We performed off-policy evaluation to all our different model settings in order to select the highest performing pedagogical policy. We compared all the different models (FC vs. LSTM, DQN vs. Double-DQN, k={1, 2, 3}) using Per-Decision Importance Sampling (PDIS) and selected the policy with the highest PDIS value for our final pedagogical policy. This method is one of the most robust off-policy evaluation methods [46]. It re-scales the rewards obtained using a behavior policy, by multiplying them by the Importance Sampling (IS) ratio ($\frac{\pi(s_i,a_i)}{\mu(s_i,a_i)}$). The IS ratio is the ratio between the probabilities given by the target policy and the behavior policy. PDIS is an alternative to regular Importance Sampling, which reduces variance in the estimations. The

PDIS formula is shown in Equation 5. In the formula, M is the number of episodes containing the state-action pair (s, a) and t_m is the first time when $(s_t, a_t) = (s, a)$ in the episode number m. γ is the discount factor and r_t is the reward at time-step t. Finally, $\pi(s_i, a_i)$ is the target policy and $\mu(s_i, a_i)$ is the behavior policy, which is random in our case. The PDIS results of the 12 models are shown in Figure 4. This evaluation was performed by training the different policies on 80% of the students, and evaluating it on the remaining 20%. The PDIS result of a random target policy is used to set $y = 0$ (the red line). Much to our surprise, while Double-DQN has shown to be much more robust in online DRL applications, its performance is generally worse than DQN here, especially when $k = 1$ and $k = 2$. Figure 4 shows that the best policy is induced using DQN with the LSTM architecture for $k = 3$, and thus is selected as our DQN policy for the empirical studies. This means that every time a decision is made, the network is using information from the last three problem states.

$$PDIS(s,a) = \frac{1}{M}\sum_{m=1}^{M}[\sum_{k=1}^{T_m - t_m}[\gamma^{k-1} r_{t_m+k} \prod_{i=t_m+1}^{t_m+k-1} \frac{\pi(s_i,a_i)}{\mu(s_i,a_i)}]] \tag{5}$$

We also compared the policy that was trained using the inferred rewards to the one that used the delayed rewards. For the comparison, we employed the Expected Cumulative Reward (ECR) value, which averages the Q-values of all the initial states to obtain an estimate of how much reward the policy will collect on average. Our train/validation split was performed using the same 80%/20% ratio as for the PDIS calculation. The policy we used for the ECR calculation was the policy that was selected for the study (DQN with LSTM layers and $k = 3$ observations). The evolution of the ECR during the training process for the inferred and delayed rewards can be seen on Figure 5.

5.4 Simple Explanations

The design of our explanation is rather straightforward. We followed the "low-controllingness" principle described in [17]. Our explanations are *action-based* in that they focused on explaining the *benefit of taking the subsequent tutorial actions*. Our simple, action-based explanations were primarily based on the prior research on learning science and cognitive science. For example, a large amount of research showed that studying WEs can be more beneficial if it is a problem involving new level of difficulty or content [41,40] and thus if the current problem was the first problem in a level, our action-based explanation for WE would state *'The AI agent thinks you should view this problem as a Worked Example to learn how some new rules work."* Our simple action-based explanation for other WE states: *"The AI agent thinks you would benefit from viewing this problem as a worked example."* Similarly, if the policy decided that the next problem should be a PS, then the message shown stated something like: *"The AI agent thinks you should solve this problem yourself."*

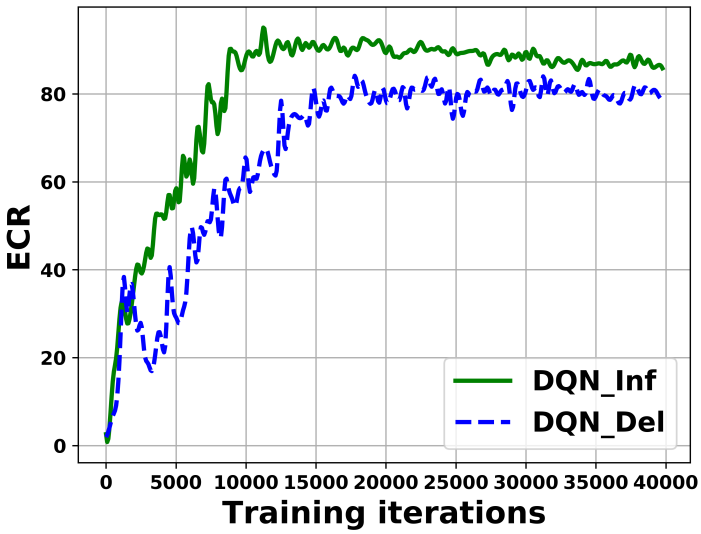

Fig. 5: Evolution of the ECR during the training process for the policies with inferred (DQN_Inf) and delayed (DQN_Del) rewards.

6 Experiment 1: Fall 2018 (F18)

6.1 Hypothesis & Experiment Setup

In this experiment, our hypothesis is that *a Batch DRL based pedagogical policy, using the DQN algorithm, will be more effective than an Expert policy.* To evaluate our hypothesis, we performed a classroom experiment with college students. A total of 84 students were randomly assigned to the two conditions using stratified sampling based on the pretest score to ensure that the two conditions had similar prior knowledge. As a result, we have $N = 41$ students for the *DQN* condition and $N = 43$ for the *Expert* baseline condition. Here the tutor in the *DQN* condition followed the induced *DQN* policies described in Section 4 *without* explanations. A one-way ANOVA test using the condition as a factor showed no significant difference in the pretest scores for the *DQN* ($M = 59.23$, $SD = 30.63$) and the *Expert* conditions ($M = 57.42$, $SD = 30.95$): $F(1, 82) = 0.07, p = 0.79$.

6.2 Learning Performance & Training Time

Learning Performance: The second column in Table 1 shows the mean (SD) posttest scores of the two conditions. Overall, no significant difference was found on the posttest between DQN and Expert-Baseline. A one-way ANCOVA analysis on posttest scores using Condition as factor and pretest

Table 1: Results of F18 study by condition.

			Log Analysis		
	PostTest	Training Time	PS Count	WE Count	Hint Count
DQN	48.6 (22.7)	60.4 (104.0)	8.5 (2.5)	7.4 (2.0)	22.8 (39.4)
Expert Baseline	54.0 (18.3)	65.8 (77.6)	7.6 (1.0)	7.6 (1.0)	15.3 (27.7)

scores as a covariate shows that there was no significant difference: $F(1, 81) = 1.76, p = 0.19$.

Training Time: The third column in Table 1 shows the mean (SD) time spent by the students in each condition during the training phase of the tutor. A one-way ANOVA test using the condition as a factor showed no significant difference on the total training time: $F(1, 82) = 0.073, p = 0.788$.

6.3 Log Analysis:

We performed a log analysis to compare the behavior of both conditions using several of the available features. We compared the number of PS and WE provided to each student by each policy, as well as the number of hints requested by the students in each condition. This analysis will help us better understand how the DQN agent behaves, and whether it favors one of the instructional interventions over the other one, and how users behaved in each condition. The reader should note that, in this analysis, the number of PS and WE the students can receive is not fixed, despite the fact that all students need to solve 20 training problems. The reason for this is that the tutor allows for a problem to be restarted or skipped, as described in Section 4.3, and the new problem can potentially be represented in a different way, following the decision made by the pedagogical policy.

The last three columns in Table 1 show the PS, WE, and hint counts of the two conditions. We performed a series of one-way ANOVA tests to measure differences in the PS, WE and hint counts, using the condition as a factor. The results showed no significant differences on the total number of WE ($F(1, 82) = 0.19, p = 0.664$), or the number of hints ($F(1, 82) = 1.019, p = 0.316$). We only found a marginal difference on the amount of PS provided by the two policies: $F(1, 82) = 3.84, p = 0.053$, where the DQN policy provided more PS than the Expert policy.

To summarize, our DQN-induced batch DRL policy did not outperform the Expert baseline policy, so we conclude that the experiment did not confirm that the DRL policy is better than the Expert policy.

7 Experiment 2: Spring 2019 (S19)

7.1 Two Hypotheses & Experiment Setup

In this study, we investigated two hypotheses: Communication Hypothesis and Agency Hypothesis. The former states that combining a Batch DRL pedagogical policy with *simple explanations (communication)* is more effective at improving student learning than an Expert-crafted policy while the latter hypothesizes that *giving students the choices (agency)* over how they want to solve each problem will be more effective than an Expert-designed policy.

To evaluate the two hypotheses, 83 students were randomly assigned to three conditions through stratified sampling: *DQN + Explanation* (*DQN+Exp*) (N = 30), *Student Choice* (N = 30), and the *Expert* baseline (N = 23). The difference in size between the two experimental conditions is due to the fact that we gave priority to have a sufficient number of participants in the explanation group to perform a meaningful analysis of how the explanations are accessed and perceived, accounting for the fact that some users might not access the explanation functionality. In the Student Choice condition, once a next problem is presented the students will make decisions on whether they want *the ITS to show* them how to solve the next problem (WE) or they want *to solve* the next problem themselves (PS). They are only required to choose at least one PS and one WE per level, which is a constraint defined by the course instructor. A one-way ANOVA test showed no significant difference in the pretest scores among the three conditions: $F(1, 81) = 0.26, p = 0.61$. More specifically, we have *DQN+Exp* ($M = 54.2$, $SD = 30.0$), *Student Choice* ($M = 50.3$, $SD = 31.3$), and *Expert* Baseline ($M = 49.9$, $SD = 35.8$). Our results suggested that all conditions were balanced in incoming competence in S19.

7.2 Learning Performance & Training Time

The S19 study had two hypotheses: the *Communication Hypothesis* is about whether DQN with simple explanations (DQN+Exp) would be more effective than the Expert baseline policy, and the *Agency Hypothesis* is about whether Student Choice can be more effective than the Expert baseline policy. In the following, we will first compare the three conditions in terms of learning performance and then perform a log analysis.

Learning Performance:

The second column in Table 2 shows a comparison of the posttest scores among the three conditions, showing the mean (and SD). A one-way ANOVA test using the condition as a factor showed a significant difference in the posttest scores: $F(1, 81) = 4.47, p = 0.037$. Furthermore, a one-way ANCOVA analysis on posttest scores using the condition as factor and the pretest scores as a covariate confirms a significant difference in the posttest scores: $F(1, 80) = 4.25, p = 0.042$.

Table 2: Results of S19 study by condition.

	PostTest	Training Time
DQN+Exp	**41.61** (25.07)	93.0 (109.6)
Student Choice	34.24 (20.09)	75.5 (104.0)
Expert Baseline	29.44 (16.43)	65.8 (87.7)

To evaluate the *Communication Hypothesis*, we compared *DQN+Exp* vs. *Expert* and contrast analysis revealed that the *DQN+Exp* condition significantly outperformed the *Expert* condition: $t(79) = 2.02, p = 0.046, d = 0.574$. For the *Agency Hypothesis*, we compared *Student Choice* vs. *Expert*, and no significant difference was found between the two: $t(79) = 0.81, p = 0.42, d = 0.261$. Additionally, no significant difference was found between the *DQN+Exp* and *Student Choice* conditions: $t(79) = 1.30, p = 0.20, d = 0.324$. In short, our results confirm the Communication Hypothesis in that on the posttest scores, *DQN+Exp* significantly outperforms the *Expert* condition, while the Agency Hypothesis was not confirmed.

Training Time: The last column in Table 2 shows the average amount of total training time (in minutes) students spent on the tutor for each condition. Despite the differences among the three conditions, a one-way ANOVA test using the condition as a factor showed no significant difference in time on task among them: $F(1, 81) = 0.97, p = 0.33$.

7.3 Log Analysis

Table 3: Log analysis of S19 study by condition.

	PS Count	WE Count	Hint Count
DQN+Exp	9.40 (2.42)	6.10 (1.21)	9.1 (10.59)
Student Choice	8.06 (3.15)	7.46 (2.14)	7.36 (8.93)
Expert Baseline	8.13 (1.74)	7.34 (1.26)	7.74 (8.42)

Table 3 shows the mean (SD) number of PSs, WEs, and hints that each condition experienced in S19. A one-way ANOVA test on the number of total PS shown, using the condition as factor, showed a marginal difference among the three conditions: $F(1, 81) = 3.54, p = 0.063$. Subsequent contrasts analyses showed a significant difference in the number of PS between the *DQN+Exp* and *Expert* conditions ($t(51) = 2.12, p = 0.038$), where the *DQN+Exp* policy provided more PS than the *Expert* policy. Furthermore, a marginal difference was found between the *DQN+Exp* and *Student Choice* conditions ($t(58) = 1.83, p = 0.071$), where the *DQN+Exp* policy provided more PS than *Student*

Choice. No significant difference was found between the *Student Choice* and the *Expert* conditions.

A one-way ANOVA test on the number of total WE shows, using the condition as a factor, showed a significant difference: $F(1, 81) = 8.32, p = 0.005$. Subsequent contrasts analyses showed a significant difference in the number of WE between the *DQN+Exp* and *Expert* conditions ($t(51) = 3.64, p < 0.001, d = 1.003$), as well as between the *DQN+Exp* and *Student Choice* conditions ($t(58) = 3.03, p = 0.003, d = 0.782$). No significant difference was found between the *Student Choice* and the *Expert* conditions.

Finally, we analyze the number of hints received by the students in each group during the training phase of the tutor. A one-way ANOVA test using the condition as a factor showed no significant difference in the number of hints shown: $F(2, 80) = 0.276, p = 0.76$.

To summarize, our log analysis shows that *DQN+Exp* generated more PS and less WE than the other two conditions and no significant difference was found between the *Student Choice* and *Expert* conditions. No significant difference was found in the number of hints received per group.

7.4 Correlation Analysis:

We performed a correlation analysis to determine whether the training time, the number of PS and WE, or the number of hints provided are significantly correlated with the posttest score.

- **Training Time and Posttest:** A Pearson's correlation test showed no significant correlation between the training time and the posttest score for all students combined: $r(81) = -0.102, p = 0.359$. When performing Pearson's correlation tests on all three conditions separately, the results showed no significant correlation for either of the conditions: $r(21) = -0.116, p = 0.596$ (Expert), $r(28) = -0.128, p = 0.499$ (Student Choice), and $r(28) = -0.141, p = 0.456$ (DQN+Exp).
- **PS count and Posttest** A Pearson's correlation test showed no significant correlation between the number of PS and the posttest score for all students combined: $r(81) = 0.126, p = 0.255$. When performing Pearson's correlation tests on all three conditions separately, the results showed no significant correlation for either of the conditions: $r(21) = -0.228, p = 0.294$ (Expert), $r(28) = 0.017, p = 0.929$ (Student Choice), and $r(28) = 0.261, p = 0.163$ (DQN+Exp).
- **WE count and Posttest** A Pearson's correlation test showed a significant correlation between the number of WE and the posttest score for all students combined: $r(81) = -0.264, p = 0.015$. When performing Pearson's correlation tests on all three conditions separately, the results only showed a significant correlation for the DQN+Exp condition: $r(21) = -0.068, p = 0.757$ (Expert), $r(28) = -0.067, p = 0.722$ (Student Choice), and $r(28) = -0.520, p = 0.003$ (DQN+Exp). The number of WEs students received is negatively correlated with their posttest scores, and

such a negative correlation is especially noticeable for DQN+Exp. It is possible that our explanations of WEs are not very effective or encouraging, leading to negative impacts on student learning as a result. It is possible, as well, that WEs were simply not an effective choice for this particular group of students, at the particular point in their learning process. Future studies must therefore investigate further on the impact of WEs and the explanations associated with them.

– **Hint count and Posttest** A Pearson's correlation test showed no significant correlation between the number of hints and the posttest score for all students combined: $r(81) = -0.144, p = 0.193$. When performing Pearson's correlation tests on all three conditions separately, the results showed no significant correlation for either of the conditions: $r(21) = -0.328, p = 0.126$ (Expert), $r(28) = -0.151, p = 0.425$ (Student Choice), and $r(28) = -0.111, p = 0.559$ (DQN+Exp).

We conclude that, overall, our analysis indicates that none of these factors are correlated with the posttest score, as we only found a significant correlation between the number of WE and the posttest score for the *DQN+Exp* condition. This further shows that it is our DQN policy combined with the simple explanations that result in higher posttest learning.

8 Conclusion, Discussion and Future Work

This work explored one potential way to combine data-driven methods such as DRL with other educational strategies that increase student autonomy and agency, and we observe that it can benefit student learning in an Intelligent Tutoring System. In this work, we investigated the impact of 1) providing students with simple explanations for the decisions of a batch DRL policy and 2) the impact of students' pedagogical decision-making on learning. We focused on whether to give students a WE or to engage them in PS. We strictly controlled the domain content to isolate the impact of the *pedagogical policy* from the *content*.

In two classroom empirical studies, we compared a batch DRL policy (with and without explanations), a Student Choice pedagogical decision making and an Expert baseline. Overall, our results detected no difference, when deciding whether to approach the next problem as PS or WE, between the batch DRL-induced policies and the Expert baseline policy, or between the Student Choice policy and the Expert baseline policy. However, by combining batch DRL-induced policies with simple explanations, we can significantly improve students' learning performance more than our Expert-designed baseline policy. One potential hypothesis is that simple explanations can promote students' buy-in to pedagogical decisions made by batch DRL induced policies. However, further survey studies are needed to determine this hypothesis. Interestingly, our study found no difference between students' problem-level decisions and the Expert baseline policy. Surprisingly, students selected as many PSs and WEs as the Expert policy, which might indicate that students don't know

which type of intervention is best for their learning process. Alternatively, it might indicate that students thought a 50/50 mix of WEs and PSs is most effective. For this reason, we think that future research should focus on inducing data-driven pedagogical policies that maximize student learning, instead of letting students choose.

We believe that the results from this research can shed some light on how to apply DRL for human-centric tasks such as ITSs. Our results indicate that students could benefit from RL-based pedagogical policies combined with explanations, and we believe these results justify further investigation and a follow-up study to reproduce our results. Furthermore, further research is required to fully understand *why* the combination of DRL and simple explanations are an effective strategy, and whether they can be applied effectively to other domains. However, in this work, we have only explored straightforward, human-expert designed explanations, which can sometimes be limiting. In the future, explainable Deep Learning techniques could be used to understand why the Neural Network has taken the decision to provide PS or WE, and measure the relevance of each state feature in taking each decision. Methods such as the permutation feature importance method could be used to determine the importance of each feature, and find out how the neural network gets impacted by the different features in the data. Better understanding of the neural network decisions could be used to build a data-driven, personalized explanation system, resulting in more powerful and accurate explanations. Furthermore, recent advances in Natural Language Processing (NLP) such as transformers allow the creation of powerful generative models that can learn to write coherent sentences and even long paragraphs. This could be used to generate adaptive, data-driven explanations that are suited to help each student succeed. To summarize all the future research directions, we devise a future where Batch DRL is used to train a policy from a past dataset of experiences; then, an explainable Neural Network algorithm explains why the DRL agent took each decision; and finally a powerful NLP system uses that information to generate explanation messages that adapt to each student's needs. This combination of algorithms could result in students feeling more engaged in the activity, while following the decisions of a policy designed to optimize their learning process.

Acknowledgements This research was supported by the NSF Grants: CAREER: Improving Adaptive Decision Making in Interactive Learning Environments(#1651909), Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments(#1726550), Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors (#2013502), Educational Data Mining for Individualized Instruction in STEM Learning Environments (#1432156).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Abdelshiheed, M., Chi, M.: Metacognition and motivation: The role of time-awareness in preparation for future learning. In: S. Denison, M. Mack, Y. Xu, B.C. Armstrong (eds.) Proceedings of the 42th Annual Meeting of the Cognitive Science Society - Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 - August 1, 2020. cognitivesciencesociety.org (2020). URL https://cogsci.mindmodeling.org/2020/papers/0167/index.html
2. Andrychowicz, M., Baker, B., et al.: Learning dexterous in-hand manipulation. arXiv preprint arXiv:1808.00177 (2018)
3. Ausin, M.S., Azizsoltani, H., Barnes, T., Chi, M.: Leveraging deep reinforcement learning for pedagogical policy induction in an intelligent tutoring system. In: Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019), vol. 168, p. 177. ERIC (2019)
4. Ausin, M.S., Azizsoltani, H., Ju, S., Kim, Y., Chi, M.: Infernet for delayed reinforcement tasks: Addressing the temporal credit assignment problem. In: Y. Chen, H. Ludwig, Y. Tu, U.M. Fayyad, X. Zhu, X. Hu, S. Byna, X. Liu, J. Zhang, S. Pan, V. Papalexakis, J. Wang, A. Cuzzocrea, C. Ordonez (eds.) 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021, pp. 1337–1348. IEEE (2021). DOI 10.1109/BigData52589.2021.9671827. URL https://doi.org/10.1109/BigData52589.2021.9671827
5. Ausin, M.S., Maniktala, M., Barnes, T., Chi, M.: Exploring the impact of simple explanations and agency on batch deep reinforcement learning induced pedagogical policies. In: I.I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, E. Millán (eds.) Artificial Intelligence in Education - 21st International Conference, AIED 2020, Ifrane, Morocco, July 6-10, 2020, Proceedings, Part I, Lecture Notes in Computer Science, vol. 12163, pp. 472–485. Springer (2020). DOI 10.1007/978-3-030-52237-7_38. URL https://doi.org/10.1007/978-3-030-52237-7_38
6. Ausin, M.S., Maniktala, M., Barnes, T., Chi, M.: Tackling the credit assignment problem in reinforcement learning-induced pedagogical policies with neural networks. In: I. Roll, D.S. McNamara, S.A. Sosnovsky, R. Luckin, V. Dimitrova (eds.) Artificial Intelligence in Education - 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14-18, 2021, Proceedings, Part I, Lecture Notes in Computer Science, vol. 12748, pp. 356–368. Springer (2021). DOI 10.1007/978-3-030-78292-4_29. URL https://doi.org/10.1007/978-3-030-78292-4_29
7. Azizsoltani, H., Kim, Y.J., Ausin, M.S., Barnes, T., Chi, M.: Unobserved is not equal to non-existent: using gaussian processes to infer immediate rewards across contexts. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 1974–1980. AAAI Press (2019)
8. Azizsoltani, H., Sadeghi, E.: Adaptive sequential strategy for risk estimation of engineering systems using gaussian process regression active learning. Engineering Applications of Artificial Intelligence **74**, 146–165 (2018)
9. Barnes, T., Stamper, J.: Automatic hint generation for logic proof tutoring using historical data. Journal of Educational Technology & Society **13**(1), 3 (2010)
10. Beck, J., Woolf, B.P., Beal, C.R.: Advisor: A machine learning architecture for intelligent tutor construction. AAAI/IAAI **2000**(552-557), 1–2 (2000)
11. Behrooz, M., Tiffany, B.: Evolution of an intelligent deductive logic tutor using data-driven elements. International Journal of Artificial Intelligence in Education **27**(1), 5–36 (2017)
12. Chi, M., Jordan, P.W., VanLehn, K.: When is tutorial dialogue more effective than step-based tutoring? In: S. Trausan-Matu, K.E. Boyer, M.E. Crosby, K. Panourgia (eds.) Intelligent Tutoring Systems - 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings, Lecture Notes in Computer Science, vol. 8474, pp. 210–219. Springer (2014). DOI 10.1007/978-3-319-07221-0_25. URL https://doi.org/10.1007/978-3-319-07221-0_25
13. Chi, M., Jordan, P.W., Vanlehn, K., Litman, D.J.: To elicit or to tell: Does it matter? In: Aied, pp. 197–204 (2009)

14. Chi, M., VanLehn, K., Litman, D., Jordan, P.: Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User Modeling and User-Adapted Interaction **21**(1-2), 137–180 (2011)
15. Cordova, D.I., Lepper, M.R.: Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. Journal of educational psychology **88**(4), 715 (1996)
16. Dabney, W., Rowland, M., Bellemare, M.G., Munos, R.: Distributional reinforcement learning with quantile regression. arXiv preprint arXiv:1710.10044 (2017)
17. Deci, E.L., Eghrari, H., Patrick, B.C., Leone, D.R.: Facilitating internalization: The self-determination theory perspective. Journal of personality **62**(1), 119–142 (1994)
18. Doroudi, S., Aleven, V., Brunskill, E.: Where's the reward? International Journal of Artificial Intelligence in Education **29**(4), 568–620 (2019)
19. Flam, J.T., Chatterjee, S., et al.: On mmse estimation: A linear model under gaussian mixture statistics. IEEE Transactions on Signal Processing **60**(7), 3840–3845 (2012)
20. Fujimoto, S., Conti, E., Ghavamzadeh, M., Pineau, J.: Benchmarking batch deep reinforcement learning algorithms. arXiv preprint arXiv:1910.01708 (2019)
21. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: International Conference on Machine Learning, pp. 2052–2062 (2019)
22. Guo, D., Shamai, S., Verdú, S.: Mutual information and minimum mean-square error in gaussian channels. arXiv preprint cs/0412108 (2004)
23. Hasselt, H.V.: Double q-learning. In: Advances in Neural Information Processing Systems, pp. 2613–2621 (2010)
24. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Applied Intelligence **31**(1), 89–106 (2009)
25. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. Knowledge-Based Systems **22**(4), 266–270 (2009)
26. Jaques, N., Ghandeharioun, A., Shen, J.H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., Picard, R.: Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. arXiv preprint arXiv:1907.00456 (2019)
27. Ju, S., Chi, M., Zhou, G.: Pick the moment: Identifying critical pedagogical decisions using long-short term rewards. In: A.N. Rafferty, J. Whitehill, C. Romero, V. Cavalli-Sforza (eds.) Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020. International Educational Data Mining Society (2020). URL https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_167.pdf
28. Kim, N., Lee, Y., Park, H.: Performance analysis of mimo system with linear mmse receiver. IEEE Transactions on Wireless Communications **7**(11) (2008)
29. Kinzie, M.B., Sullivan, H.J.: Continuing motivation, learner control, and cai. Educational Technology Research and Development **37**(2), 5–14 (1989)
30. Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education (IJAIED) **8**, 30–43 (1997)
31. Kohn, A.: Choices for children. Phi Delta Kappan **75**(1), 8–20 (1993)
32. Kumar, A., Fu, J., Soh, M., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: Advances in Neural Information Processing Systems, pp. 11784–11794 (2019)
33. Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: Reinforcement learning, pp. 45–73. Springer (2012)
34. Laroche, R., Trichelair, P., Des Combes, R.T.: Safe policy improvement with baseline bootstrapping. In: International Conference on Machine Learning, pp. 3652–3661. PMLR (2019)
35. Mandel, T., Liu, Y.E., Levine, S., Brunskill, E., Popovic, Z.: Offline policy evaluation across representations with applications to educational games. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pp. 1077–1084. International Foundation for Autonomous Agents and Multiagent Systems (2014)

36. Maniktala, M., Cody, C., Barnes, T., Chi, M.: Avoiding help avoidance: Using interface design changes to promote unsolicited hint usage in an intelligent tutor. International Journal of Artificial Intelligence in Education **30**(4), 637–667 (2020)
37. Maniktala, M., Cody, C., Isvik, A., Lytle, N., Chi, M., Barnes, T., et al.: Extending the hint factory for the assistance dilemma: A novel, data-driven helpneed predictor for proactive problem-solving help. Journal of Educational Data Mining **12**(4), 24–65 (2020)
38. McLaren, B.M., van Gog, T., et al.: Exploring the assistance dilemma: Comparing instructional support in examples and problems. In: Intelligent Tutoring Systems, pp. 354–361. Springer (2014)
39. McLaren, B.M., Isotani, S.: When is it best to learn with all worked examples? In: AIED, pp. 222–229. Springer (2011)
40. McLaren, B.M., Isotani, S.: When is it best to learn with all worked examples? In: International Conference on Artificial Intelligence in Education, pp. 222–229. Springer (2011)
41. McLaren, B.M., Lim, S.J., Koedinger, K.R.: When and how often should worked examples be given to students? new results and a summary of the current state of research. In: Proceedings of the 30th annual conference of the cognitive science society, pp. 2176–2181 (2008)
42. Minsky, M.: Steps toward artificial intelligence. Proceedings of the IRE **49**, 8–30 (1961)
43. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
44. Najar, A.S., Mitrovic, A.: Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. UMUAI **26**(5), 459–491 (2016)
45. Newell, A., Simon, H.A., et al.: Human problem solving, vol. 104. Prentice-Hall Englewood Cliffs, NJ (1972)
46. Precup, D., Sutton, R.S., Singh, S.P.: Eligibility traces for off-policy policy evaluation. In: ICML, pp. 759–766. Citeseer (2000)
47. Rafferty, A.N., Brunskill, E., et al.: Faster teaching via pomdp planning. Cognitive science **40**(6), 1290–1332 (2016)
48. Rasmussen, C.E.: Gaussian processes in machine learning. In: Summer School on Machine Learning, pp. 63–71. Springer (2003)
49. Renkl, A., Atkinson, R.K., et al.: From example study to problem solving: Smooth transitions help learning. The Journal of Experimental Education **70**(4), 293–315 (2002)
50. Rowe, J.P., Lester, J.C.: Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. In: AIED, pp. 419–428. Springer (2015)
51. Salden, R.J., Aleven, V., Schwonke, R., Renkl, A.: The expertise reversal effect and worked examples in tutored problem solving. Instructional Science **38**(3), 289–307 (2010)
52. Schraw, G., Flowerday, T., Reisetter, M.F.: The role of choice in reader engagement. Journal of Educational Psychology **90**(4), 705 (1998)
53. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning, pp. 1889–1897 (2015)
54. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
55. Schwab, D., Ray, S.: Offline reinforcement learning with task hierarchies. Machine Learning **106**(9-10), 1569–1598 (2017)
56. Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Aleven, V., Salden, R.: The worked-example effect: Not an artefact of lousy control conditions. Computers in Human Behavior **25**(2), 258–266 (2009)
57. Shen, S., Ausin, M.S., Mostafavi, B., Chi, M.: Improving learning & reducing time: A constrained action-based reinforcement learning approach. In: UMAP, pp. 43–51. ACM (2018)
58. Shen, S., Chi, M.: Aim low: Correlation-based feature selection for model-based reinforcement learning. International Educational Data Mining Society (2016)

59. Shen, S., Chi, M.: Reinforcement learning: the sooner the better, or the later the better? In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, pp. 37–44. ACM (2016)
60. Shen, S., Mostafavi, B., Lynch, C., Barnes, T., Chi, M.: Empirically evaluating the effectiveness of pomdp vs. mdp towards the pedagogical strategies induction. In: International Conference on Artificial Intelligence in Education, pp. 327–331. Springer (2018)
61. Shyu, H.Y., Brown, S.W.: Learner control versus program control in interactive videodisc instruction: What are the effects in procedural learning. International Journal of Instructional Media **19**(2), 85–95 (1992)
62. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
63. Silver, D., Hubert, T., Schrittwieser, J., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
64. Stamper, J., Barnes, T., Lehmann, L., Croy, M.: The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track, pp. 71–78 (2008)
65. Sweller, J., Cooper, G.A.: The use of worked examples as a substitute for problem solving in learning algebra. Cognition and Instruction **2**(1), 59–89 (1985)
66. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI, vol. 2, p. 5. Phoenix, AZ (2016)
67. VanLehn, K., Graesser, A.C., et al.: When are tutorial dialogues more effective than reading? Cognitive science **31**(1), 3–62 (2007)
68. Vinyals, O., Babuschkin, I., Czarnecki, W., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**, 350 (2019)
69. Wang, P., Rowe, J., Min, W., Mott, B., Lester, J.: Interactive narrative personalization with deep reinforcement learning. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (2017)
70. Yeh, S.W., Lehman, J.D.: Effects of learner control and learning strategies on english as a foreign language (efl) learning from interactive hypermedia lessons. Journal of Educational Multimedia and Hypermedia **10**(2), 141–159 (2001)
71. Zhou, G., Azizsoltani, H., Ausin, M.S., Barnes, T., Chi, M.: Hierarchical reinforcement learning for pedagogical policy induction. In: International conference on artificial intelligence in education, pp. 544–556. Springer (2019)
72. Zhou, G., Azizsoltani, H., Ausin, M.S., Barnes, T., Chi, M.: Hierarchical reinforcement learning for pedagogical policy induction (extended abstract). In: C. Bessiere (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 4691–4695. ijcai.org (2020). DOI 10.24963/ijcai.2020/647. URL https://doi.org/10.24963/ijcai.2020/647