Student-Initiated Action Advising via Advice Novelty

Ercument Ilhan, Jeremy Gow and Diego Perez-Liebana

***Abstract*—Action advising is a budget-constrained knowledge exchange mechanism between teacher-student peers that can help tackle exploration and sample inefficiency problems in deep reinforcement learning (RL). Most recently, student-initiated techniques that utilise state novelty and uncertainty estimations have obtained promising results. However, the approaches built on these estimations have some potential weaknesses. First, they assume that the convergence of the student's RL model implies less need for advice. This can be misleading in scenarios with teacher absence early on where the student is likely to learn suboptimally by itself; yet also ignore the teacher's assistance later. Secondly, the delays between encountering states and having them to take effect in the RL model updates in presence of the experience replay dynamics cause a feedback lag in what the student actually needs advice for. We propose a student-initiated algorithm that alleviates these by employing Random Network Distillation (RND) to measure the novelty of a piece of advice. Furthermore, we perform RND updates only for the advised states to ensure that the student's own learning does not impair its ability to leverage the teacher. Experiments in GridWorld and MinAtar show that our approach performs on par with the state-of-the-art and demonstrates significant advantages in the scenarios where the existing methods are prone to fail.**

***Index Terms*—Reinforcement learning (RL), deep q-network (DQN), action advising, teacher-student.**

I. Introduction

DEEP reinforcement learning (RL) has achieved outstanding feats in a wide range of domains including board games [1], complex video games [2][3] and robotics [4] in the recent years. In addition to its remarkable performance, deep RL's end-to-end structure and general applicability make it a desirable decision-making tool for many real-world problems. Nevertheless, such advantages come at the cost of long training times and large numbers of samples acquired through environment interactions, which are well-known drawbacks of deep RL [5]. There is currently a significant amount of research effort focused on improving exploration techniques to make agents more proficient at discovering the environment mechanics and collecting samples that will yield high-reward strategies [6]. Alternatively, in some situations, agents may also have access to legacy knowledge of some peers, be it humans or other agents. This presents a whole new set of opportunities to tackle the exploration challenges by leveraging these.

Peer-to-peer knowledge transfer techniques have been investigated in various forms in RL to accelerate the learning processes [7]. The applications within deep RL can be examined in two main groups as follows: learning from demonstration (LfD) [8] and action advising [9]. In LfD, the agent has access to some previously collected data by one [10] or more [11] peers to be leveraged in a pre-training stage or during the actual RL stage to improve its learning performance. Action advising, however, does not deal with the previously collected data, and instead, is concerned with devising techniques for the situations where the agent has access to a competent peer that can provide advice in the form of actions upon request. This is similar to the active learning paradigm in supervised learning [12]. As the main focus of this study, action advising concept is especially important in the scenarios where it is costly or not possible at all to generate useful expert data in advance, e.g., when the relevant task specifications are unknown a priori.

The initial problem setup of the action advising framework [9] assumed the learning agent, namely, the student, to be monitored constantly by a teacher who is responsible to manage the distribution of these advice. However, this is impractical because of possible limitations in communication and the teacher's attention span. Moreover, the teacher has no access to the student's internal model, which renders its advising decisions to be determined solely by the teacher's initiative. Therefore, this framework has also been extended into different versions such as student-initiated and jointly-initiated, letting the student take an active role in initiating these interactions, as covered in Section II. Despite its potential advantages, it is far from trivial to devise efficient student-initiated action advising strategies in deep RL, as the limited number of the studies may suggest.

In deep RL, assigning credit to a transition for its long-term contribution to the learning progress is a very challenging task. Accordingly, it is also difficult to determine the actual importance of a state to obtain expert advice for. Therefore, the student-initiated approaches in deep RL currently follow heuristics as proxies of potential learning contribution, such as novelty [13][14] and uncertainty [15][16] estimations. Despite demonstrating promising performances by simplifying challenging characteristics of deep RL, these techniques have several drawbacks. First, these estimations become smaller in correlation with the student's RL model convergence, which means that the student becomes less likely to request advice for the states it has already encountered. Even though this behaviour is intended, it may be problematic in some scenarios. For instance, the teacher may not be accessible early in the training due to communication related issues (that can occur in real-world domains) or because it joins the training at some later time just like in a multi-agent system. Moreover, if the teacher also has a non-stationary policy, it may refuse to give any advice in some stages of the learning session, which can be considered as absence. In this case, despite the learning progression of the student's model, it will lack an adequate amount of expert contribution and will face the risk of remaining suboptimal.

Consequently, even when the teacher becomes available at a later timestep, it will be ignored by the student due to the aforementioned property of these approaches; hence, the student will fail to benefit from the advice to improve its policy further. Apart from this weakness, if the student's learning algorithm incorporates an experience replay mechanism as in off-policy RL, e.g., Deep Q-Network (DQN) [17], it takes some time between encountering a state and involving it in the model update steps for a sufficient number of times to have it influence the state novelty and uncertainty estimations. As result, these delays may mislead the student about what advice it actually needs to collect, especially since the contents of its replay memory is disregarded by these action advising methods.

In this study, we first highlight and demonstrate some drawbacks of the existing student-initiated action advising methods. Then, we propose a state novelty based method involving Random Network Distillation (RND) [18], which, unlike the previous studies, is exclusively updated for the states that are advised (hence the name *advice novelty*). In other words, the mechanism that drives the decisions of the student's advice queries gets to be affected by only these states. Thus, it is ensured that the student will not ignore the teacher regardless of its own state of learning unless it has already utilised the teacher's advice.

Finally, RND updates are performed with single samples in an on-line fashion instead of with batches off a replay memory. This prevents the RND model from converging to a global optima, which gives the earlier states a chance to be asked for advice again periodically. This is similar to the idea of keeping expert transitions in the replay memory over the course of learning to be benefited from occasionally, as in [10]. Because, no matter how many times the expert actions are executed, there is no guarantee for the expert behaviour demonstrated in them to be completely distilled into the student's policy. This way of RND learning minimises the possibility of the advice to be discarded forever without providing enough contribution. Additionally, this can be advantageous if the teacher happens to have a non-stationary policy that generates advice actions. In this case, getting advised by the teacher in different stages of learning will yield different outcomes, which is essential to learn from everything the teacher has to offer.

This paper is structured as follows: Section II provides a review of the related work. Then, the relevant background information are provided in Section III. The game environments used in this study as presented in Section IV. In Section V, we describe our proposed method in detail. The experiment setup is explained in Section VI. Afterwards, the results are presented and discussed in Section VII. Finally, the study is concluded with some final remarks in VIII.

II. Related Work

Action advising was formalised in [9] for the first time as teacher-student framework. According to this, a student agent is observed constantly by a teacher agent to be given action advice in order to make it achieve the best possible learning performance. However, the number of advice requests is limited with a budget constraint due to practical concerns. The problem of distributing this budget is addressed with several heuristics they proposed, such as early advising and importance advising. Following the proposed ideas, several different approaches in this line of work have emerged. In [19], the teacher-initiated action advising problem was treated as a meta-level RL problem. By training the student agent over multiple learning sessions, the teacher tries to learn the most appropriate times to advise by using the student's learning state as reward feedback. Another study [20] extended the initially proposed heuristic-based idea to student-initiated and jointly-initiated forms by devising several new heuristics. In [21], the case of having multiple teachers is investigated, as well as dealing with situations where the teacher's expertise may be suboptimal. Using RL to achieve optimal advising was also studied in [22] similarly to [19]. As well as learning *when* to advise, the teachers are also made to learn *what* to advise in this work. [13] applied heuristic-based action advising to the domain of multi-agent RL with cooperative agents. The agents are made to determine their expertise by counting state visits to take on student or teacher roles dynamically to exchange knowledge with their peers to improve team-wide learning.

Deep RL is a relatively new application domain for action advising methods. [23] is one of the first studies in this particular subject. Even though it was based on the agents with classical RL methods, the teaching via action advising problem is tackled as a meta-level deep RL problem to determine the actions to be advised as well as their temporal distribution. Similarly, [24] further developed this idea to also be able to handle agents that employ deep RL. Despite being effective in forming teacher-student interactions bidirectionally without any fixed roles, these approaches share the drawbacks of requiring the agents to be trained over multiple, centralised learning sessions due to the meta-RL mechanisms. This may also cause the agents to end up being tuned for each other and face difficulties when paired with different peers.

Another line of work in deep RL involves applying action advising methods by following heuristics without any pre-training, which forms the specific group of approaches our study belongs to. In [16], the LfD paradigm was combined with active learning by making the student agent query for demonstrations itself in a very similar way to action advising. To do so, they make use of some specialised network architectures that make it possible for the student agent to measure its state uncertainty, and be able to use this estimation to determine the most appropriate times and states to request advice for. [14] further developed the idea of [13] to make the agents initiate teacher-student interactions in multi-agent deep RL. Instead of state counting, RND is employed to compute the novelty of the states to be compatible with complex state spaces that require non-linear function approximation. More recently, [15] utilised uncertainty measurements obtained via multiple neural network heads in student-initiated action advising to time the advice requests similarly to [16]; however, they don't employ a special loss function as in LfD. In parallel to these studies, [25] introduced an approach to imitate and reuse the teacher advice in order to prevent spending budget on previously queried states, which can be combined with other action advising methods to make them more practically feasible.

In addition to the aforementioned drawbacks, none of these heuristic-based methods for deep RL is designed to be able to handle the extended absence of the teacher. As we highlight in Section I, this is especially crucial in real-world applications. Our study aims to provide a solution that is suitable for such scenarios, as well as alleviating the present individual drawbacks of requiring uncertainty models [15][16], being susceptible to experience replay induced latency in estimations [14]–[16] which are detailed in Section V.

III. Background

A. Reinforcement Learning and Deep Q-Networks

Reinforcement learning (RL) is a prominent framework for solving sequential decision-making tasks that involve an agent acting in an environment that is formalised by a *Markov decision process* (MDP). In an MDP, an environment is defined by a tuple $\langle S, A, R, T\rangle$, where S is finite set of states, A is finite set of actions, $R\colon S \times A \times S \to \mathbb{R}$ is reward function and $T\colon S \times A \to \Delta(S)$ is transition function. Acting in an MDP happens according to the agent's policy $\pi\colon S \to A$ which maps states to actions. At each timestep t, the agent observes the state s_t and executes the action a_t in order to receive the reward r_t and advance to the next state s_{t+1}. The objective of RL algorithms is to obtain the π^* that maximises expected sum of discounted rewards $\sum_{k=0}^{T} \gamma^k r_{t+k}$ obtained from timestep t over a horizon T ($\gamma \in [0, 1]$ is the discount factor).

Deep Q-Network (DQN) [17] is a scaled-up version of the well known fundamental RL technique Q-learning with non-linear function approximation suitable for complex domains, which attempts to obtain an optimal policy by learning state-action values $Q(s, a)$ in an end-to-end fashion. To do so, it utilises a neural network with weights θ to map an input state s to $Q(s, a)$ by minimising the loss term $(r_{k+1} + \gamma \max_{a'} Q_{\bar{\theta}}(s_{k+1}, a') - Q_\theta(s_k, a))^2$ via stochastic gradient descent over randomly sampled transitions from timestep k. Furthermore, DQN also utilises certain mechanisms to achieve functionality. One of these is keeping the network weights θ in a separate copy network that is updated periodically and is used as a reference point in updates to stabilise learning. Another critical component of DQN is called experience replay. It stands for the process of saving the encountered transitions in a replay memory to be used in the model updates later on. Not only such off-policy updates improve sample efficiency, but also this mechanism helps to break the non-i.i.d. property of the samples to make convergence easier.

Following its breakthrough, DQN is enhanced with additional techniques over years that are combined under the name of Rainbow DQN [26]. In our study, we employ double Q-learning [27], dueling networks [28] and noisy networks [29] among these, which we believe are the most essential ones. Despite being a substantial enhancement, Prioritised Experience Replay [30] is not utilised in our work, since it may amplify the effects of having more important samples in the replay memory, and consequently make the comparison between different action advising methods less impartial. Hence, uniformly random sampling is chosen as the default experience replay approach.

B. Noisy Networks

Noisy Networks (NoisyNets) [29] are an eminent exploration technique for deep RL algorithms, which is also employed in Rainbow DQN [26] as the default exploration strategy. In principle, NoisyNets is a modified linear layer with noise perturbations as follows:

$$y = (\mu_w + \sigma_w \odot \epsilon_w)x + \mu_b + \sigma_b \odot \epsilon_b \tag{1}$$

where the input is $x \in \mathbb{R}^p$, the output is $y \in \mathbb{R}^q$, the learnable parameters are $\mu_w \in \mathbb{R}^{q\times p}$, $\sigma_w \in \mathbb{R}^{q\times p}$, $x \in \mathbb{R}^p$, $\mu_b \in \mathbb{R}^q$, $\sigma_b \in \mathbb{R}^q$, and the noise parameters are $\epsilon_w \in \mathbb{R}^{q\times p}$, $\epsilon_b \in \mathbb{R}^q$ for input size of p and output size of q. As the learning progresses, the noisy components are ignored in a state-conditional trend, giving the agent an implicitly driven exploration ability.

Another important benefit of having NoisyNets is having access to a form of uncertainty estimation by using the predictive variance of a noisy layer as described in [16]. This is computed for action a and state s in the final layer that outputs $Q(s, a)$ values as follows:

$$\begin{aligned} Var[Q(s,a;\theta)] &= Var[w_a \phi(s)] + Var[b_a] \\ &= \phi(s)^\intercal diag({\sigma_{w_a}}^2)\phi(s) + {\sigma_{b_a}}^2 \end{aligned} \tag{2}$$

where $\phi(s)$ is the latent features generated from state s to be fed this layer, w_a and b_a are the weight and bias terms, respectively. The uncertainty in state s is then obtained by taking the predictive variance of the action with largest Q-value as follows:

$$U(s) = Var[Q(s, \arg\max_a Q(s,a))] \tag{3}$$

Even though our proposed method does not utilise uncertainty measurements, we employ NoisyNets in the student agent to make it possible to compare our approach with uncertainty-based ones as well.

C. Random Network Distillation

Random Network Distillation (RND) [18] is proposed to address difficult exploration problems by providing the agent with an intrinsic state novelty bonus. It uses two randomly initialised neural networks, *target* and *predictor*, denoted by G and $\hat{G}$. Over the course of learning, the *predictor* network $\hat{G}$ is updated to minimise the mean squared error $\|\hat{G}(s) - G(s)\|^2$ between the embeddings of these two networks, for every state s used to update the RL algorithm's model. As the states are encountered in the learning, this error becomes smaller for such states as the predictor network becomes better at matching the target network's output. Thus, this error term can be used as a measurement of novelty for any state.

D. Action Advising

Action advising [9] is a form of inter-agent knowledge transfer method with a very simple mechanism. By using only a common understanding over a communication protocol and a set of actions, a teacher exchanges instructions in form of actions with a student agent in order to improve its learning

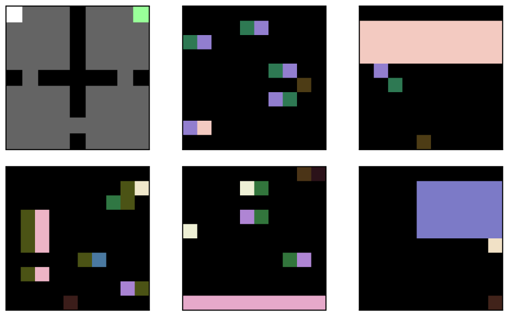

Fig. 1: Rendered observations of the initial state of GridWorld (top-left) and random states from MinAtar games Asterix (top-center), Breakout (top-right), Freeway (bottom-left), Seaquest (bottom-center) and Space Invaders (bottom-right).

process by biasing its interactions. An important aspect of this framework is the limitation of the number in these interactions with a budget, considering the setbacks of real-world use cases. For instance, the communication with the teacher may be costly, or its attention may be limited as in the case of a human teacher. Furthermore, even with an available budget, the teacher can not be assumed to be available at all times.

While some of the approaches use simpler heuristics to distribute these advice, more advanced ones in deep RL employ techniques that involve estimating state novelty [14] or uncertainty [15][16] from the student agent's perspective. Depending on the method, either of the peers can be initiating the knowledge transfer process. In this study, we follow the student-initiated approach and use the following baselines as well as state uncertainty and state novelty based ones:

- **Early advising:** By intuition, the agents are thought to benefit more from teaching earlier in the learning process, which forms the main motivation of this heuristic [9]. According to this, the peers request/receive advice constantly from the beginning until they run out of budget.
- **Random advising:** Despite the advantages of acquiring action advice early on, an agent may still discover some unfamiliar states later in the training as it learns. Therefore, it may also be helpful to distribute the available budget over a longer period of time. This heuristic provides an uninformed way of scattering these by making the peers determine to initiate action advising with some probability (usually 0.5) at each timestep until the budget is consumed.

IV. The Game Environments

In this study, we employ two different game environments, namely, GridWorld and MinAtar, to represent different aspects of learning challenges.

A. GridWorld

GridWorld[1] is a grid structured environment with a size of 9×9, composed of ground (grey), pit (black) tiles, and goal (green) as visualised in Figure 1 (left) where the agent is represented in white. The environment is perceived by the agent as binary tensors with a size of $9 \times 9 \times 3$. The agent starts in a fixed position in the top-left corner of the top-left room (as shown in a white cell in Figure 1, top-left), and must navigate to the goal tile in the top-right corner of the top-right room, with 4 available actions in order to get a reward of $+1$. Every time the agent attempts to take an action, it has a chance to execute another action uniformly at random instead, with a probability of 0.1; however, if a random action happens to move the agent on a pit tile, it instead holds the agent in its previous position. Reaching the goal, stepping on a pit tile or exceeding the maximum timesteps of 50 terminates the episode with 0 points of reward. Considering the possibility of random movements and the tight timesteps limit, this game presents a significant exploration challenge despite its simple rules.

B. MinAtar

MinAtar [31] is an environment composed of minimal versions of five popular Atari games[2]. While the core game mechanics are kept the same, the observation and the action spaces are reduced to cut down the representational complexity. Every game has a common action space of 6, and the observations are binary tensors with sizes of $10 \times 10 \times n$, where n denotes the number of the object categories in the game. Different from the default version, we set any game episode to have a maximum timesteps of 1000. Figure 1 shows screenshots taken from these games, and their brief descriptions are as follows:

- **Asterix:** The game has a constant stream of enemies and treasures that spawn and move in horizontal directions. The agent can move in 4 directions and must collect treasures to obtain $+1$ reward while avoiding enemies which kills the agent and terminates the game.
- **Breakout:** The agent controls a paddle at the bottom of the screen by moving it in horizontal directions. The objective is to keep hitting the ball to keep it within the screen to avoid losing the game. Hitting the bricks with the ball breaks them and yields $+1$ reward. New lines of bricks keep appearing after they are cleared up.
- **Freeway:** The agent's objective is to cross the way by avoiding the cars moving at different speeds in horizontal directions. Upon reaching the goal, $+1$ reward is awarded and the agent is sent back to the starting point, and the cars' directions and speeds are randomised. Being hit by a car also sends the agent back to the starting point.
- **Seaquest:** This is the game with the most complex rules among MinAtar games. The agent controls a submarine in a sea filled with enemy submarines, fish, and divers to be rescued by navigating around. Shooting the enemies yields $+1$ reward, as well as taking each diver to the surface. Moreover, the agent must keep an eye on the remaining oxygen level and have it replenished by going to the surface; which increases the difficulty each time it is done. If the agent goes up to the surface with no divers, is hit, or has no remaining oxygen, the game terminates.
- **Space Invaders:** The agent is in charge of controlling a space ship that can shoot bullets to the upcoming group

[1]Codes are available at https://github.com/ercumentilhan/GridWorld

[2]Our version: https://github.com/ercumentilhan/MinAtar/tree/original

of aliens from the top of the screen. Each shot down alien yields a reward of $+1$, and upon being cleared up, a new wave of aliens spawns with an increased movement speed. The aliens can also shoot back at the agent. If the agent is hit by a bullet or an alien, the game is terminated.

Directions of the moving sprites are also encoded in the observations by having a separate category for their trails to ensure full observability. Asterix, Seaquest and Space Invaders also involve a periodical difficulty ramping that occurs at every 100^{th} timestep.

V. Action Advising via Advice Novelty

In our problem setting, we follow the standard RL framework and MDP formalisation as in Section III-A. Our setup considers a situation where a student agent that employs an off-policy deep RL algorithm, e.g., DQN, with policy π_S is learning to excel in a given task. There is also a peer who is competent in this task to be treated as a teacher with a fixed policy π_T. The student can access this peer and sample actions from π_T for a limited number of times determined by the action advising budget b. The objective of the agent is to utilise an action advising procedure to distribute the available advice requesting budget b in the best possible way to maximise its learning performance.

The samples obtained in deep RL are not only used for discovering the environment but are also responsible for driving the learning of the agent's model. Additionally, every single step of environment interaction and model update influences the sample collection process right after. These make it very difficult for deep RL agents to predict what kind of long-term effects a transition, or a single piece of advice in this case, will have on their learning progress. Therefore, student-initiated action advising techniques rely on simple heuristics and proxies of expected usefulness of samples in terms of learning contribution. These, however, are prone to fail in several ways.

Early advising is a strong baseline due to the fact that the samples obtained early on have more influence on deep RL algorithms' learning progress. However, it lacks the ability to distribute the available advising budget in more critical states. This is likely to result in the budget being wasted, and make the agent miss important advice opportunities especially when the budget is small. Additionally, having no stopping condition in making advice requests may cause the agent to get over-advised, which can deteriorate the learning performance as we show in Section VII. Employing another common heuristic, uniformly random advising, can alleviate this drawback. Nonetheless, not being able to follow the teacher's policy consistently causes this method to be unsuccessful in the tasks with sparse rewards that require deep exploration, as also shown in Section VII.

The more advanced techniques that rely on state importance surrogates such as novelty [13][14], model uncertainty [16][15], perform better in general. Though, they still have drawbacks that can be problematic in some cases.

First of all, updates in these estimations are driven by the student's learning progression, regardless of having any teacher contribution in it. If the teacher is present from the beginning, the student's RL model convergence can be assumed to be towards well-explored state-value targets with the support of teacher advice. Otherwise, if the teacher remains absent for a significant amount of time as described in Section I, there can be no guarantees for the student to explore the environment successfully. As we show in Section VII, it is indeed very likely for the student to end up with a converged suboptimal policy even in a simple task with sparse rewards. In this case, the student will ignore the teacher regardless of its potential knowledge to be leveraged.

Secondly, based on the student's deep RL model's properties, there may be some delays between the advice collection and its actual value to be taken effect in the model, e.g., off-policy updates with replay memory where the collected samples are held in a buffer and are employed periodically to update the model. This lag prevents the student from being precise about what advice it actually needs. Even though we do not demonstrate this behaviour in our experiments, we ensure there can be no model induced delays in estimations that affect our action advising approach.

Finally, these approaches require several restrictions on the student's RL algorithm. On one hand, the state novelty-based approach needs to have access to the batches of samples the students use to update its learning model. This prevents the student from remaining as a blackbox. On the other hand, uncertainty-based methods require the student's model to be capable of providing a notion of uncertainty, which is possible only in a small subset of RL algorithms.

In order to address these shortcomings, we propose *action advising via advice novelty*, a method that employs state novelty measurements to time advice requests. In our technique, the student agent employs an RND module that consists of two randomly initialised neural networks G and $\hat{G}$ with identical structures. At each step with available advice requesting budget, the agent measures novelty n_s of the state s it encounters as the RND loss $n_s = \|\hat{G}(s) - G(s)\|^2$. This value is then converted into a linearly decreasing advice requesting probability as $p_s = n_s/\eta$ (clipped to be in $[0, 1]$), where η is a predefined threshold. If the advice request takes place, then $\hat{G}$ is updated to minimise the loss term $\|\hat{G}(s) - G(s)\|^2$. Thus, n_s becomes smaller as the agent receives advice for s. This can then be seen as a novelty metric for a piece of advice to be obtained in a particular state, considering the assumption of teacher policy π_T being fixed, and ignoring the environment's stochasticity. By performing updates only when advised, it is ensured that the student always attempts to learn from the teacher, no matter how far into convergence its task-level model is. Furthermore, the RND updates occur right after the student is advised with only a single piece of observation rather than a batch of them. This prevents the RND module to minimise its loss globally, and cause it to have relatively high novelty for the states it has not encountered in a while, giving these states a chance to be re-advised. Finally, since RND employs neural networks with non-linear function approximation, our method can function in complex domains and is capable of generalising between unseen states. A full description of our method can be seen in Algorithm 1.

Algorithm 1 Action Advising via Advice Novelty

```
1: Input: action advising budget $b$, agent policy $\pi_S$, teacher policy $\pi_T$, novelty threshold $\eta$
2: for training steps $t \in \{1, 2, \dots k\}$ do
3:     get observation $s_t \sim Env$ if $Env$ is reset
4:     $a_t \leftarrow None$
5:     if $b > 0$ then
6:         $n_{s_t} \leftarrow \|\hat{G}(s_t) - G(s_t)\|^2$          ▷ compute novelty
7:         $p_{s_t} \leftarrow n_{s_t}/\eta$          ▷ compute probability
8:         sample $p$ uniformly at random in $[0, 1]$
9:         if $p_{s_t} > p$ then
10:            $a_t \sim \pi_T$          ▷ request advice
11:            Update $\hat{G}$ weights to minimise $\|\hat{G}(s) - G(s)\|^2$
12:            $b \leftarrow b - 1$
13:        end if
14:    end if
15:    if $a_t$ is $None$ then
16:        $a_t \sim \pi_S$
17:    end if
18:    Execute $a_t$ and obtain $r_t$, $s_{t+1} \sim Env$
19:    Update the RL model, e.g., DQN.
20:    $s_t \leftarrow s_{t+1}$
21: end for
```

VI. Experimental Setup

We are interested in investigating the shortcomings of the existing student-initiated action advising approaches, and evaluating how our approach compares with them in different scenarios. For this purpose, we choose GridWorld and MinAtar games to conduct experiments in two stages involving different challenges. We compare the following modes of student agents with different action advising approaches:

- **No Advising (None)**: The agent does not employ any form of action advising; it follows its own policy at all times.
- **Early Advising (EA)**: The agent follows the early advising heuristic to distribute its action advising budget by requesting advice until it runs out of its action advising budget.
- **Random Advising (RA)**: The agent follows the random advising heuristic and determines whether to request a piece of advice uniformly at random at each step until it runs out of its action advising budget.
- **Uncertainty-Based Advising (UA)**: Advice requests are made according to the student's RL model uncertainty, similarly to [15], [16]. Specifically, at each step, the student's NoisyNets uncertainty is obtained for the current state, and then is divided by a threshold ν to determine advice requesting probability.
- **State Novelty-Based Advising (SNA)**: Advice requests are driven by state novelty measurements, similarly to [14]. For each state the student encounters, its novelty is measured by a separate RND module. Then, this value is divided by a predefined threshold ρ to obtain advice requesting probability. The RND module is updated simultaneously with the student's RL model, by the same batches of samples.
- **Advice Novelty-Based Advising (ANA)**: The agent that follows our proposed approach.

TABLE I: Hyperparameters of DQN (top section) and RND (bottom section).

Hyperparameter name	Value	
	GridWorld	MinAtar
Replay memory size to start learning	1000	10000
Replay memory capacity	10000	100000
Target network update period	250	1000
Minibatch size	32	32
Learning rate	0.0001	0.0001
Train period	2	2
Huber loss δ	1	1
Learning rate	0.0001	0.0001
Normalisation steps	1k	5k

The student agent's RL algorithm is set to be DQN, including the prominent extensions of double Q-learning, dueling networks, and NoisyNets exploration strategy. The neural network structure is comprised of a single convolutional layer consisting 16 3×3 filters with a stride of 1, followed by a fully connected noisy layer with 128 hidden units. The RND module employs a similar neural network structure with a single convolutional layer consisting 16 3×3 filters with a stride of 1, followed by a regular fully connected layer with 128 hidden units and 6 output units (set arbitrarily). These networks do not share any weights or sample batches used during the training. The hyperparameters of DQN and RND are tuned for each environment prior to the experiments (can be seen in Table I) and the discount factor is set as 0.99 in all experiments.

As described in the problem setup, there needs to be a teacher for the student agent to be able to get advice from. In GridWorld, we set our teacher policy as following the shortest path from the current position to the goal tile. In MinAtar, we generated competent teachers by training separate DQN agents for each of the games for 3M steps, who achieve final evaluation scores (as defined later on) of 29.28, 81.15, 5.77, 146.64, 146.06 in Asterix, Breakout, Freeway, Seaquest, Space Invaders, respectively. The teachers are made to employ ϵ-greedy exploration instead of NoisyNets, to have them as dissimilar as possible from the student in order to eliminate any possible advantages that may arise in the knowledge exchange process due to them being identical.

Every student variant (e.g., None, EA, etc.) is trained for a fixed number of steps which we define as a learning session. During a learning session, the agent is evaluated periodically in a separate sequence of episodes with having any form of exploration (e.g., the noise perturbations of NoisyNets) and teaching disabled. The scores obtained in these episodes are averaged to determine the evaluation score for this evaluation step. These scores reflect the agent's actual expertise in the corresponding step in the learning session. In GridWorld, since the actual cumulative rewards at the end of an episode is either 0 or 1, a more informative evaluation score is defined to be in $[0, 1]$ by taking the number of remaining timesteps and distance to the goal tile into calculations. In MinAtar, the original game scores in the framework are used as evaluation scores. In addition to the evaluation scores, we also plot the number of advice taken in every 100 steps as well as cumulatively, to observe the trends in budget spending.

In the first stage of experiments, we use GridWorld as a simple and interpretable task to highlight the aforementioned drawbacks, as well as performing a preliminary benchmark on the methods to validate their suitability for tasks with more complex mechanics. The learning session lengths are set as 100k steps, and evaluations are performed at every 100^{th} step in a sequence of 5 episodes. We set two different scenarios, namely, Scenario I and Scenario II, with small and large budget options of 5k and 50k each, resulting in 4 cases in total. In Scenario I, the teacher is present from the beginning of learning sessions, which is the common experimental setting used in the previous action advising studies. In Scenario II, the teacher joins the loop at the 25kth step. By having such a scenario, we test the ability of the student in dealing with the belated teacher. UA, SNA, ANA hyperparameters ν, ρ, η are determined empirically in the 5k budget setting, to be 0.001, 0.0001, 0.001 and are kept the same for all 4 cases.

In the second stage, we evaluate the approaches in a set of tasks with more complex dynamics presented via 5 different games in MinAtar environment. Learning sessions are set to have a length of 1.5M steps and evaluations are performed at every 1000^{th} step in a sequence of 5 episodes. Since the Scenario II experiments in GridWorld are sufficient to demonstrate the weakness regarding the extensive unavailability of the teacher, experiments in MinAtar are only conducted for Scenario I to evaluate the general performance of the methods, again with two different budget options of 50k and 250k. UA, SNA, ANA hyperparameters ν, ρ, η are determined empirically on game by game basis for 50k budget, to be 0.0001, 0.0025, 0.001 for Asterix; 0.001, 0.025, 0.025 for Breakout; 0.0001, 0.1, 0.05 for Freeway; 0.001, 0.05, 0.05 for Seaquest; 0.0025, 0.01, 0.0025 for Space Invaders, respectively.

Experiment results are aggregated over 9 different seeds for GridWorld, and over 20 different seeds for MinAtar games. Training and evaluation episode sequences (random game events) are fixed via random seeds and kept the same across different experiment seeds. Therefore, these experiment seeds only affect the agents' internal computations. Finally, we perform RND observation normalisation as in [18] by using the mean and standard deviation calculated over the first 1000 and 5000 observations, in GridWorld and MinAtar, respectively[3].

VII. Results and Discussion

The results of our experiments are presented in Figure 2 (GridWorld); and in Figures 3 and 4, and Table II (MinAtar). In Figure 2, the leftmost column contains two plots for the number of advice taken in total and in every 100 steps, in Scenario I with the budget amount of 50k; the middle column displays the evaluation scores in Scenario I, and the rightmost column displays the evaluation scores in Scenario II (with the budgets of 5k on top and 50k on bottom). Figure 3 includes the plots of the number of advice taken in total and in every 100 steps, in Scenario I with the budget amount of 250k for all five MinAtar games. In Figure 3, plots of evaluation scores obtained in MinAtar games with two different budget settings of 50k (top row) and 250k (bottom row) are displayed. These results are presented numerically in Table II with the area under the curve and final values. The table also shows whether the results are significantly different than ANA's results according to Welch's t-test with $p < 0.05$. $(+)$ sign on the left-hand side of a result indicates that ANA is significantly better than the corresponding method. Similarly, $(-)$ indicates that ANA is significantly worse than it. We also denoted the best results in their own brackets in bold. The plots in Figure 2 and Figure 3 that display the number of advice taken are only generated for Scenario I with the maximum budget settings since the budget distribution is identical in the cases with smaller budgets with only the difference of being cut off early. The curves are plotted with appropriate moving average smoothing for the sake of comprehensibility, and the standard deviation across the runs are shown with the shaded areas.

A. GridWorld

In Scenario I with a small budget of 5k, all of the action advising methods performed reasonably well, with the exception of RA which fails to be any better than None. The poor performance of None indicates how challenging it can be to conduct exploration successfully even with an advanced method like NoisyNets when the time constraints are tight as in this GridWorld game. Despite taking plenty of expert advice, RA also fails due to its inability to consistently follow the teacher, which is especially essential in the tasks requiring deep exploration like this one. This makes RA an unreliable action advising heuristic.

When the budget is increased to 50k, we see how the performances collapse due to taking too much advice hence not executing their own policies adequately to collect integral samples. This is most obvious in EA since it employs the most greedy way of spending the budget amongst all methods, which makes it dangerously susceptible to such high budget settings. With the addition of more budget, RA finally manages to benefit from expert advice, however still very inadequately. The advanced methods of UA, SNA, and our ANA do a good job on not overusing all the budget given to them even though they are not tuned to handle 50K. While UA performs slightly worse, ANA does better with the addition of an extra budget. As it can be seen, ANA does this while using more budget in the end than UA; this shows that it is not just about cutting off the advice requests but is also about distributing them in the appropriate states and across the learning session. In terms of budget efficiency, SNA seems to be doing the best in this case; however, also has worse task performance. Finally, differently than UA and SNA, ANA is observed to follow a trend with occasional peaks in the number of advice taken per 100 steps. This is very likely to be caused by its RND update rule that uses single samples rather than batches that are also non-i.i.d. when the advice requests are made consecutively. As a result, RND model does not achieve global optimum and remain to

[3]Codes for our experiments can be found at https://github.com/ercumentilhan/advice-novelty

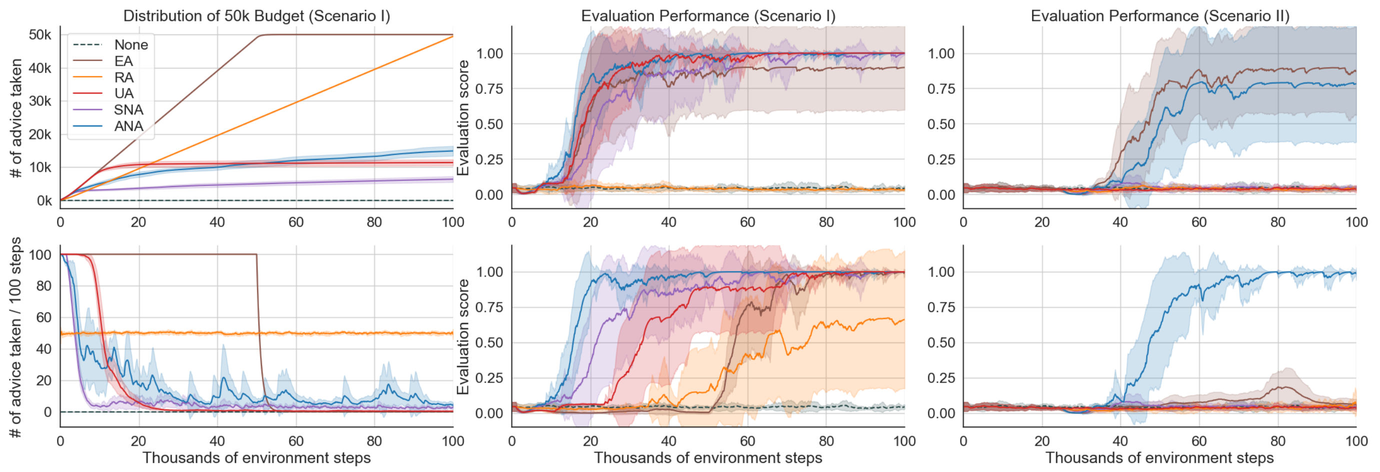

Fig. 2: Number of advice taken cumulatively and in every 100 steps period in Scenario I with 50k budget (leftmost column), evaluation scores in Scenario I (middle column) and Scenario II (rightmost column) with 5k (top row) and 50k (bottom row) budgets, obtained in GridWorld game with no action advising (None) and action advising methods EA, RA, UA, SNA, ANA.

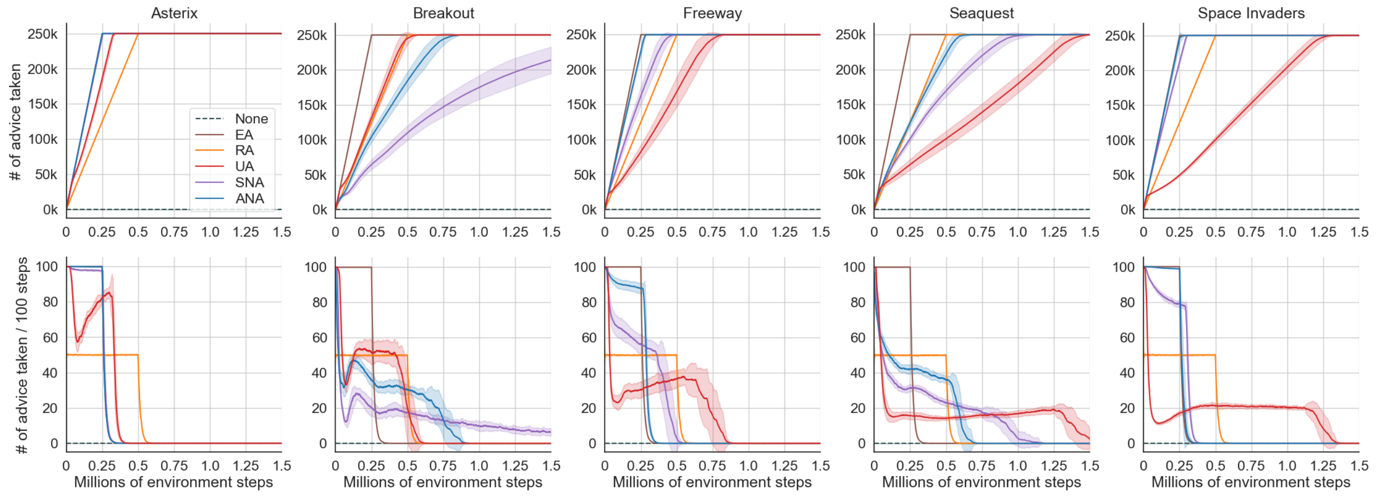

Fig. 3: Number of advice taken cumulatively and in every 100 steps period in Scenario I with 250k budget, obtained in five MinAtar games with no action advising (None) and action advising methods EA, RA, UA, SNA, ANA.

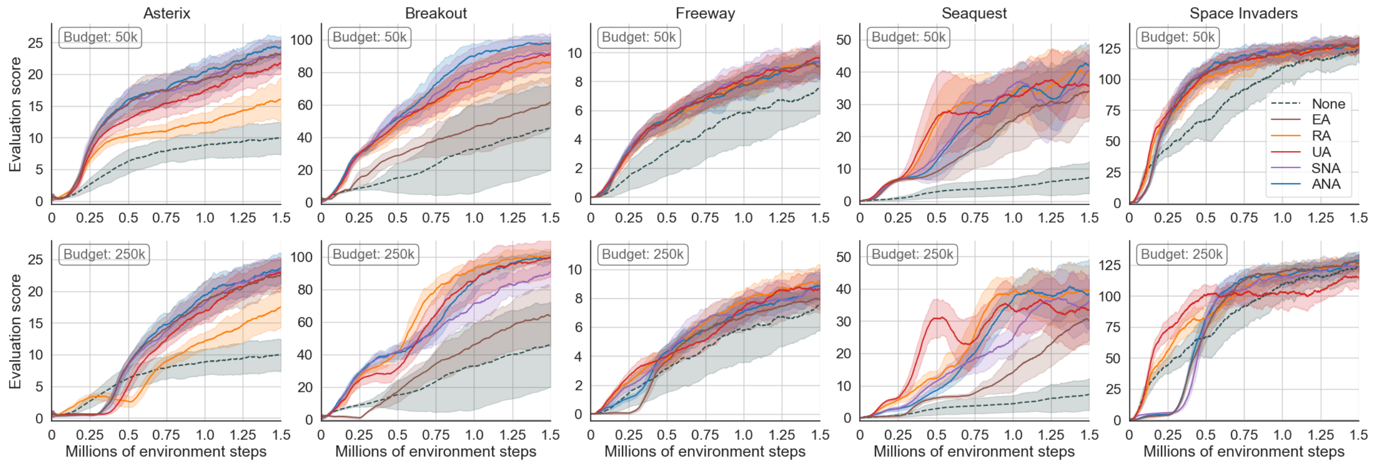

Fig. 4: Evaluation scores of Scenario I with 50k (top row) and 250k (bottom row) budgets, obtained in five MinAtar games with no action advising (None) and action advising methods EA, RA, UA, SNA, ANA.

TABLE II: Area under the curve (AUC) and the mean of last 50 values (Final Score) of the evaluation score plots of None, EA, RA, UA, SNA, ANA agent modes obtained in 5 MinAtar games averaged over 20 runs. The numbers denoted by ± represent standard deviation. The percentage values in parentheses indicate the relative difference to the values obtained by EA. *Overall* section in bottom presents these percentages averaged over these 5 games. (+) and (−) on the left-hand side of the results indicate whether ANA's results are significantly better or worse than them, respectively (according to Welch's t-test with $p < 0.05$). The best results in their own brackets are denoted in bold.

		Final Score		AUC ($\times 10^3$)	
Game	Mode	50k	250k	50k	250k
Asterix	None	(+) 9.91 ± 2.4 (−57.2%)	(+) 9.91 ± 2.4 (−55.7%)	(+) 10.01 ± 2.2 (−57.1%)	(+) 10.01 ± 2.2 (−45.2%)
	EA	23.16 ± 2.0	22.37 ± 2.4	23.33 ± 2.1	18.28 ± 2.1
	RA	(+) 15.83 ± 2.9 (−31.7%)	(+) 17.16 ± 3.0 (−23.3%)	(+) 15.67 ± 1.5 (−32.9%)	(+) 12.62 ± 1.4 (−31.0%)
	UA	(+) 21.57 ± 1.4 (−6.9%)	22.79 ± 2.3 (+1.9%)	(+) 20.77 ± 1.7 (−11.0%)	(+) 16.83 ± 1.4 (−7.9%)
	SNA	23.00 ± 2.2 (−0.7%)	23.04 ± 1.9 (+3.0%)	23.25 ± 2.6 (−0.3%)	18.29 ± 1.9 (+0.02%)
	ANA	**24.24 ± 1.6(+4.7%)**	**23.36 ± 2.3(+4.4%)**	**24.19 ± 1.6(+3.7%)**	**19.02 ± 1.6(+4.0%)**
Breakout	None	(+) 45.30 ± 25.5 (−25.1%)	(+) 45.30 ± 25.5 (−28.9%)	(+) 35.94 ± 22.2 (−32.2%)	(+) 35.94 ± 22.2 (−24.1%)
	EA	(+) 60.48 ± 15.7	(+) 63.73 ± 18.1	(+) 52.98 ± 12.0	(+) 47.34 ± 10.6
	RA	(+) 86.06 ± 12.2 (+42.3%)	**100.65 ± 3.6(+57.9%)**	(+) 83.73 ± 9.7 (+58.1%)	(−) **97.32 ± 3.4(+105.6%)**
	UA	(+) 90.45 ± 7.1 (+49.6%)	98.84 ± 11.5 (+55.1%)	(+) 86.81 ± 7.0 (+63.9%)	88.02 ± 9.8 (+86.0%)
	SNA	(+) 91.50 ± 11.6 (+51.3%)	(+) 89.32 ± 7.4 (+40.1%)	(+) 90.67 ± 10.3 (+71.2%)	(+) 80.58 ± 6.2 (+70.2%)
	ANA	**97.78 ± 4.0(+61.7%)**	99.12 ± 3.7 (+55.5%)	**97.38 ± 5.9(+83.8%)**	89.83 ± 4.3 (+89.8%)
Freeway	None	(+) 7.36 ± 1.6 (−19.3%)	(+) 7.36 ± 1.6 (−7.3%)	(+) 6.25 ± 2.2 (−31.1%)	(+) 6.25 ± 2.2 (−10.0%)
	EA	9.12 ± 0.8	(+) 7.94 ± 0.9	9.08 ± 0.6	(+) 6.95 ± 0.6
	RA	9.30 ± 0.9 (+2.0%)	**9.09 ± 1.1(+14.4%)**	9.03 ± 0.7 (−0.5%)	(−) **8.37 ± 0.7(+20.5%)**
	UA	**9.60 ± 1.0(+5.2%)**	8.59 ± 1.2 (+8.2%)	9.28 ± 0.7 (+2.2%)	7.97 ± 0.7 (+14.7%)
	SNA	9.29 ± 1.2 (+1.9%)	8.61 ± 1.0 (+8.4%)	**9.38 ± 0.8(+3.4%)**	7.98 ± 0.7 (+14.8%)
	ANA	9.33 ± 1.0 (+2.2%)	8.82 ± 0.9 (+11.1%)	9.06 ± 0.6 (−0.2%)	7.58 ± 0.6 (+9.1%)
Seaquest	None	(+) 7.06 ± 4.6 (−79.1%)	(+) 7.06 ± 4.6 (−76.7%)	(+) 5.33 ± 3.1 (−79.4%)	(+) 5.33 ± 3.1 (−68.4%)
	EA	(+) 33.76 ± 8.8	(+) 30.36 ± 7.3	(+) 25.84 ± 6.2	(+) 16.85 ± 3.6
	RA	39.38 ± 8.2 (+16.6%)	**39.37 ± 7.4(+29.7%)**	(−) **37.10 ± 5.4(+43.6%)**	(−) 35.33 ± 3.6 (+109.6%)
	UA	(+) 35.98 ± 9.6 (+6.6%)	33.83 ± 9.8 (+11.4%)	(−) 36.60 ± 5.2 (+41.6%)	(−) **36.82 ± 4.0(+118.5%)**
	SNA	39.29 ± 6.3 (+16.4%)	35.19 ± 8.0 (+15.9%)	33.46 ± 4.0 (+29.5%)	(+) 28.07 ± 2.7 (+66.6%)
	ANA	**42.21 ± 6.0(+25.0%)**	38.57 ± 9.6 (+27.1%)	32.77 ± 4.6 (+26.8%)	31.72 ± 3.0 (+88.2%)
Space Invaders	None	122.62 ± 9.4 (−3.8%)	(+) 122.62 ± 9.4 (−3.8%)	(+) 123.41 ± 13.2 (−17.3%)	123.41 ± 13.2 (−0.4%)
	EA	**127.45 ± 6.8**	127.40 ± 6.2	149.19 ± 5.2	123.86 ± 4.7
	RA	125.95 ± 3.9 (−1.2%)	124.91 ± 5.2 (−2.0%)	146.72 ± 5.9 (−1.7%)	(−) **137.96 ± 4.7(+11.4%)**
	UA	126.56 ± 6.0 (−0.7%)	(+) 115.38 ± 8.7 (−9.4%)	150.67 ± 5.8 (+1.0%)	(−) 135.39 ± 7.4 (+9.3%)
	SNA	127.40 ± 5.4 (−0.04%)	124.95 ± 6.6 (−1.9%)	**150.82 ± 5.2(+1.1%)**	(+) 120.99 ± 6.3 (−2.3%)
	ANA	127.23 ± 5.5 (−0.2%)	**128.26 ± 6.6(+0.7%)**	150.14 ± 6.4 (+0.6%)	125.05 ± 4.8 (+1.0%)
Overall	None	−36.9%	−34.5%	−43.4%	−29.6%
	RA	+5.6%	+15.3%	+13.3%	+43.2%
	UA	+10.8%	+13.4%	+19.5%	**+44.1%**
	SNA	+13.8%	+13.1%	+21.0%	+29.9%
	ANA	**+18.7%**	**+19.8%**	**+22.9%**	+38.4%

yield significantly higher loss for the sample(s) that are not encountered recently. This is a unique characteristic of ANA which can be advantageous as it makes the teacher advice to be re-acquired occasionally.

In Scenario II, where the teacher joins the learning session at 25kth step, both UA and SNA fail to learn from the teacher as expected, due to their teacher independent convergence in estimations of uncertainty and novelty, respectively. Our method ANA, however, manages to leverage the teacher's knowledge in both budget options, despite of the student being converged to suboptimal Q-value targets due to under-exploration. Clearly, if there is such a possibility as depicted in this scenario, methods like UA and SNA are not going to be suitable action advising methods to be employed.

B. MinAtar

Results in MinAtar games presents us a more general performance evaluation of the action advising techniques in a variety of tasks. The final evaluation scores are what we mainly consider when comparing the algorithms. Yet, we also analyse the AUC values to assess their performance in terms of learning speed.

We first discuss the results obtained in the 50k budget setting as the primary comparison case, since the algorithms are tuned particularly for this setup. In Asterix, ANA is the best performing method with SNA and EA being very close to it, and they are followed by UA and RA. Considering the inefficacy of RA and the successful budget spending patterns of EA, SNA and ANA; it can be speculated that it is critical in this particular game to follow the teacher over many consecutive steps, similarly to GridWorld. In Breakout, ANA outperforms other advanced methods which already do well compared to None and EA. This in comparison to the case in Asterix shows how EA and RA can be unreliable choices as a trade-off for being very simple heuristics. Unlike GridWorld and Asterix, the successful methods in Breakout are the ones that ask for advice less frequently; this may be due to Breakout not requiring expert advice over many steps since the game events unfold on their own once the ball is hit with the paddle, and the different random moves taken in the meantime may be much more valuable sources to reduce RL model error, rather than taking the same expert advised actions such as just waiting stationarily until the ball traverses back down. Seaquest has very interesting results where every method achieves different standings in different stages of the learning session. In terms of the final performance, however, ANA manages to come on top again with SNA and RA following it after. UA performs rather poorly here, despite its rapid progression earlier in the learning. As a result of the Seaquest's in-game dynamics being the most complicated amongst all, it is not very clear what causes the learning fluctuations in these plots. In Freeway and Space Invaders, despite all the different advice requesting patterns followed by different methods, they are very similar when it comes to the evaluation scores. Even though it may be surprising at first considering that Freeway is a game with sparse rewards, its action space and the possible positions the agent can traverse in the game grid spatially are rather small. Therefore, the need for the expert advice to be distributed strategically across the game episode is rather negligible.

When the budget is increased to 250k, we observe a fair amount of performance deterioration in almost every advising mode, especially in the learning speeds. In Asterix, Freeway, and Space Invaders, even though the final performance is not affected greatly, there is a sharp drop in the learning progression caused by the over-advising induced delay in the collection of useful samples. This is closely linked to them behaving more similarly to EA in these cases as it can be seen in the budget plots. These results emphasise the importance of handling the redundant advice budgets. Currently, none of these action advising methods have an awareness of how many times they will get to ask for advice. Instead, they are designed to make the most out of some supposedly small budget they are given, without any notion of long-term planning of its utilisation.

Overall, as the performance superiority of the advising methods over None suggests, action advising provides substantial advantages to accelerate learning. By looking at the standings of the algorithms in every game, we can see that our ANA is the winner in terms of the final performance both in the 50k and 250k budget options; it either achieves the top scores or remain very close to them. This is also visible in the overall percentages of final score improvements over EA, in which ANA achieved 18.7% and 19.8% while its closest followers got 13.8% (SNA) and 15.3% (RA), respectively in 50k and 250k budget settings. The significance analyses also support ANA's superiority by showing that it significantly outperformed its competitors in 23 out of 60 cases. Depending on the budget, ANA's performance is followed by the other methods in a different order. For instance, while SNA is far ahead of others and is behind ANA in 50k budget, RA takes its place in the 250k budget scenario. The decline in performance with higher budgets and RA's robustness to this by spacing out the advice requests to allow the student to execute its self policy more often points out the importance of collecting on-policy samples adequately even when the agent itself employs an off-policy RL algorithm, as highlighted in [32][33] as well. Clearly, different games require different strategies to distribute the action advising budget, and there is no straightforward way to determine a way that applies to all cases successfully. Furthermore, the ways these methods behave are also dependent on the underlying task, since the uncertainty and novelty estimations may be affected differently even with the identical streams of observations. For instance, while UA tends to spend its budget more slowly than others in most cases, in Breakout it behaves differently to output its best possible performance. Finally, it is worth it to mention that ANA is also observed to stand out in terms of computational efficiency compared to the runner-up SNA; while ANA updates its RND model with a single sample only when it successfully receives advice, SNA updates it for a batch of samples every time it performs learning until its budget reaches zero.

VIII. Conclusions

In this work, we evaluated the prominent student-initiated action advising methods that are compatible with off-policy deep RL agents; and highlighted their shortcomings such as not being able to handle the belated availability of the teacher, or requiring very specific underlying algorithms to work. To address these, we proposed an alternative student-initiated action advising algorithm that utilises state novelty computed via Random Network Distillation (RND) to determine when to request a piece of advice. Differently from the previous works, RND is updated only with the states that are involved in the advice exchanges. Thus, it is ensured that the student will take advantage of the teacher as soon as it becomes available regardless of its own RL model's convergence.

Empirical results in GridWorld and MinAtar games validate our speculations of the aforementioned drawbacks and show that the state-of-the-art methods that utilise state novelty or uncertainty can be ineffective if the teacher is not present from the beginning. Furthermore, our advice novelty approach manages to be the strongest among its competitors by yielding the best overall standings in the majority of the experiments, as well as being able to handle belated teachers, not requiring deep RL model uncertainty estimations, and also not interfering with the student's RL algorithm. It is also seen that there is no trivial way to define a general action advising strategy to distribute the budget efficiently across many different cases. Finally, it is found to be challenging for even the most complicated methods to handle excessive budgets without encountering a significant performance deterioration. Accordingly, the hyperparameters that are responsible to manage budget distribution require careful tuning considering both the task characteristics and the total available budget.

An interesting extension of this study would be further investigating the components of our approach to precisely determine how they behave and how their variants affect the performance. For instance, training RND incrementally with batches drawn from the complete set of collected advice instead of only the latest samples can provide valuable insights for comparison. Another direction for future work could involve devising a form of threshold adaptation to make the action advising techniques more robust against the changes in task and budget specifications. Additionally, further analyses in the dynamics of different RL algorithms operating with action advising would be imperative to invent more general action advising methods. Our study employs a student agent with DQN and NoisyNets exploration; it will be worthwhile to investigate the performance of the action advising algorithms with different RL algorithms and exploration methods to see how their standings vary. Finally, there is still a significant research gap in action advising with multiple teachers which requires further attention, and we believe that our approach is likely to be useful in such a problem setting.

Acknowledgment

This research utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT. http://doi.org/10.5281/zenodo.438045

References

[1] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017.

[2] O. Vinyals, I. Babuschkin, J. Chung, *et al.*, *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*, https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

[3] C. Berner, G. Brockman, B. Chan, *et al.*, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019.

[4] S. Levine, C. Finn, T. Darrell, *et al.*, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, 39:1–39:40, 2016.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *CoRR*, vol. abs/1708.05866, 2017.

[6] A. A. Taıga, W. Fedus, M. C. Machado, *et al.*, "Benchmarking bonus-based exploration methods on the arcade learning environment," *CoRR*, vol. abs/1908.02388, 2019.

[7] F. L. da Silva, G. Warnell, A. H. R. Costa, and P. Stone, "Agents teaching agents: A survey on inter-agent transfer learning," *Auton. Agents Multi Agent Syst.*, vol. 34, no. 1, p. 9, 2020.

[8] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, M. Mozer, M. I. Jordan, and T. Petsche, Eds., MIT Press, 1996, pp. 1040–1046.

[9] L. Torrey and M. E. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, 2013, pp. 1053–1060.

[10] T. Hester, M. Vecerık, O. Pietquin, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 3223–3230.

[11] D. Seita, D. M. Chan, R. Rao, *et al.*, "ZPD teaching strategies for deep reinforcement learning from demonstrations," *CoRR*, vol. abs/1910.12154, 2019.

[12] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[13] F. L. da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2017, May 8-12, 2017*, ACM, 2017, pp. 1100–1108.

[14] E. Ilhan, J. Gow, and D. Pérez-Liébana, "Teaching on a budget in multi-agent deep reinforcement learning," in *IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019*, 2019, pp. 1–8.

[15] F. L. da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 5792–5799.

[16] S. Chen, V. Tangkaratt, H. Lin, and M. Sugiyama, "Active deep q-learning with demonstration," *Mach. Learn.*, vol. 109, no. 9-10, pp. 1699–1725, 2020.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[18] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by random network distillation," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[19] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: A reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[20] O. Amir, E. Kamar, A. Kolobov, and B. J. Grosz, "Interactive teaching strategies for agent training," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, S. Kambhampati, Ed., IJCAI/AAAI Press, 2016, pp. 804–811.

[21] Y. Zhan, H. Bou-Ammar, and M. E. Taylor, "Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 2315–2321.

[22] A. Fachantidis, M. E. Taylor, and I. P. Vlahavas, "Learning to teach reinforcement learning agents," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 21–42, 2019.

[23] S. Omidshafiei, D. Kim, M. Liu, *et al.*, "Learning to teach in cooperative multiagent reinforcement learning," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2019, pp. 6128–6136.

[24] D. Kim, M. Liu, S. Omidshafiei, *et al.*, "Learning hierarchical teaching in cooperative multiagent reinforcement learning," *CoRR*, vol. abs/1903.03216, 2019.

[25] E. Ilhan, J. Gow, and D. Perez-Liebana, "Action advising with advice imitation in deep reinforcement learning," in *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, ACM, 2021, pp. 629–637.

[26] M. Hessel, J. Modayil, H. van Hasselt, *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), February 2-7, 2018*, 2018, pp. 3215–3222.

[27] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, D. Schuurmans and M. P. Wellman, Eds., AAAI Press, 2016, pp. 2094–2100.

[28] Z. Wang, T. Schaul, M. Hessel, *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, M. Balcan and K. Q. Weinberger, Eds., ser. JMLR Workshop and Conference Proceedings, vol. 48, JMLR.org, 2016, pp. 1995–2003.

[29] M. Fortunato, M. G. Azar, B. Piot, *et al.*, "Noisy networks for exploration," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[30] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[31] K. Young and T. Tian, "Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments," *CoRR*, vol. abs/1903.03176, 2019.

[32] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 2052–2062.

[33] A. Kumar, J. Fu, *et al.*, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 11 761–11 771.