Learning on a Budget via Teacher Imitation

Ercüment İlhan, Jeremy Gow and Diego Perez-Liebana
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
{e.ilhan, jeremy.gow, diego.perez}@qmul.ac.uk

Abstract—Deep Reinforcement Learning (RL) techniques can benefit greatly from leveraging prior experience, which can be either self-generated or acquired from other entities. Action advising is a framework that provides a flexible way to transfer such knowledge in the form of actions between teacher-student peers. However, due to the realistic concerns, the number of these interactions is limited with a budget; therefore, it is crucial to perform these in the most appropriate moments. There have been several promising studies recently that address this problem setting especially from the student's perspective. Despite their success, they have some shortcomings when it comes to the practical applicability and integrity as an overall solution to the learning from advice challenge. In this paper, we extend the idea of advice reusing via teacher imitation to construct a unified approach that addresses both advice collection and advice utilisation problems. We also propose a method to automatically tune the relevant hyperparameters of these components on-the-fly to make it able to adapt to any task with minimal human intervention. The experiments we performed in 5 different Atari games verify that our algorithm either surpasses or performs on-par with its top competitors while being far simpler to be employed. Furthermore, its individual components are also found to be providing significant advantages alone.

Index Terms—reinforcement learning, deep q-networks, action advising, teacher-student framework

I. Introduction

Deep Reinforcement Learning (RL) has been proven to be a successful approach to solve decision-making problems in a variety of difficult domains such as video games [1], board games [2] or robot manipulation [3]. However, achieving the reported performances is not entirely straightforward. One of the most critical setbacks in deep RL is the exhaustive training processes that usually require many interactions with the environment. This occurs mainly due to the RL inherent exploration challenges as well as the complexity of the incorporated function approximators, e.g. deep neural networks. To this date, there has been a remarkable amount of research effort to overcome the sample inefficiency by devising advanced exploration strategies [4]. In addition to these, other lines of work that focus on leveraging some legacy knowledge to tackle these issues have also been studied extensively with great success.

The ability to learn by utilising the prior experience instead of starting from scratch is an essential component of intelligence. In RL, this idea has been investigated in various forms that are tailored for different problem settings. Imitation Learning (IL) [5] studies the concept of mimicking an expert behaviour presented in a pre-recorded dataset without allowing any RL rewards from the environment itself. Similarly, Learning from Demonstrations (LfD) [6][7] extends this idea to incorporate the RL interactions and rewards to further surpass the experts. Some other approaches such as Policy Reuse [8] study the ways of speeding up the agent's learning by leveraging the past policies directly, instead of datasets.

In this paper, we study a different problem setup where it is not possible to have any pre-generated datasets or to directly obtain the useful policies themselves; instead, the learning agent only has access to some peer(s) over a limited communication channel. Such a setting is especially relevant for the scenarios with unknown task specifications (prior to learning and deployment) and non-transferable yet beneficial policies, e.g. humans, non-stationary agents, in the loop. A flexible framework that is tailored for this setting is called Action Advising [9]. According to this, agents exchange knowledge between each other in the form of actions to speed-up their learning progressions. However, the number of these peer-to-peer interactions are limited with a budget to resemble the real-world limitations, which essentially converts the problem into determining the best possible way to utilise the available budget. Based on the peer that is in charge of driving these interactions, action advising algorithms can either be student-initiated, teacher-initiated or jointly-initiated.

Action advising algorithms in deep RL have obtained promising results with an emphasis on student-initiated strategies [10][11][12]. While the majority of these focus on addressing *when to ask for advice* question, some recent approaches [13] have also investigated the ways to further utilise the collected advice by imitating and reusing the teacher policy. Despite these developments, there are currently several significant shortcomings present. These techniques often employ some threshold hyperparameters to control the decisions to initiate advice exchange interactions which play a key role in their efficiencies. However, these parameters are sensitive to the learning state of the models as well as the domain properties. Therefore, they need to be tuned very carefully prior to execution, which involve unrealistically accessing the target tasks for trial runs. Furthermore, the studies to further leverage the teacher advice beyond collection are currently in their early stages and do not provide a complete solution to the problem besides addressing the advice reusing aspect.

In this paper, we present an all-in-one student-initiated

approach that is capable of collecting and reusing advice in a budget-efficient manner, by extending [13] in multiple ways. First, we propose a method for automatically determining the threshold parameters responsible for the decisions to request and reuse advice. This greatly alleviates the burden of task-specific hyperparameter tuning procedures. Secondly, we follow a decaying advice reuse schedule that is not tied to the student's exploration strategy. Finally, instead of using the imitated policy only for reusing advice as in [13], we incorporate this policy to determine and collect more diverse advice to construct a more universal imitation policy.

The rest of this paper is structured as follows: Section II outlines the most relevant previous work. In Section III, the background knowledge that is required to understand the paper is provided. Afterwards, we describe our approach in detail in Section IV. Then, Section V presents our evaluation domain and experimental procedure. In Section VI, we share and analyse the experiment results. Finally, Section VII wraps up this study with conclusions and future remarks.

II. RELATED WORK

Action advising techniques with budget constraints were originally invented and studied extensively in tabular domains. In [9], the teacher-student learning procedure was formalised for the first time and some solutions from the teacher's perspective were proposed. This was later extended with some new heuristics [14] as well as with a meta-learning approach [15]. Later on, the action advising interactions are also studied in student-initiated and jointly-initiated forms [16] which were also adopted in multi-agent problems where the agents take student and teacher roles interchangeably [17]. In a recent work [18], the idea of reusing the previously collected advice is investigated to improve the learning performance and also to make a more efficient use of the available budget.

Deep RL is a considerably new domain for the action advising studies. [19] introduced a novel LfD setup in which the demonstrations dataset is built interactively as in action advising. To do so, they employed uncertainty estimation capable models with LfD loss terms integrated in the learning stage. [10] extended the jointly-initiated action advising [17] to be applicable in multi-agent deep RL for the first time. This study replaced the state counters that were used to assess uncertainty in the tabular version [17] with state novelty measurements with the aid of Random Network Distillation (RND) [20]. In [11], an uncertainty-based advice collection strategy was proposed. According to this, the student adopts a multi-headed neural network architecture to access epistemic uncertainty estimations as in [19]. Later, [12] further studied the state novelty-based idea of [10] to devise a better student-initiated advice collection method. Specifically, they made RND module to be updated only for the states that are involved in advice collection. That way, the student was ensured to benefit from the teacher regardless of its own knowledge about the states to tackle the special cases of belated inclusion of the teacher. More recently, [13] studied the advice reuse idea in deep RL. In this work, an imitation model of the teacher is constructed partially with the collected advice. Furthermore, Dropout regularisation [21] is also incorporated in this model to enable it to make uncertainty-aware decisions when it comes to reusing the self-generated advice.

Among these studies in deep RL, only [19] and [13] investigated the concept of advice reuse. Our paper differs from them in several ways. The idea in [19] require using uncertainty estimation capable models in the student's RL algorithm. Moreover, the RL algorithm also needs to be modified to have the LfD loss terms. In contrast, the algorithm we propose does not require the student to have any specific RL models or loss functions. Thus, the student agent can be treated as a blackbox, which lets our method to be applied to a wider range of agent types. [13] performs advice reuse as a separate module, as we described here. However, they rely on the previously proposed advice collection strategies instead of taking advantage of its own imitation module to manage the advice collection process via uncertainty. Furthermore, some of the hyperparameters in [13] limit its practical applicability due to being difficult to tune, which we also address in this work.

III. BACKGROUND

A. Reinforcement Learning and Deep Q-Networks

Reinforcement Learning (RL) [22] is a trial-and-error learning paradigm that studies decision-making problems where an agent learns to accomplish a task. This interaction within the environment is commonly formalised as a Markov Decision Process (MDP). In MDP, the environment is defined with the tuple $\langle \mathcal{S}, \mathcal{A}, R, \mathcal{T}, \gamma \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of available actions, $R(s, a)$ is the reward function, $\mathcal{T}(s|a, s')$ defines the transition probabilities, and $\gamma \in [0, 1]$ is the discount factor. At every timestep t, the agent applies an action a_t in state s_t to advance to the next state s_{t+1} while receiving a reward r_t. These actions are determined by the agent's policy $\pi \colon \mathcal{S} \to \mathcal{A}$, and RL's goal is to learn an agent policy π_θ (with parameters θ) that maximises the cumulative discounted sum of the rewards $\sum_{t=0}^{\infty} \gamma^t r_t$. There are various different approaches in the RL literature to achieve this. For instance, the well-known Q-learning algorithm does so by learning the state-action values $Q(s, a)$ via Bellman equations [22] and making the agent follow the policy $\pi(s) = \operatorname{argmax}_a Q(s, a)$.

In the recent years, RL algorithms have been studied extensively in a branch referred to as Deep RL to deal with non-tabular state spaces with the aid of non-linear function approximation. Deep Q-Networks (DQN) [23] is a substantial one among these, that serves as a strong baseline in the domains with discrete actions. In this off-policy algorithm, $Q(s, a)$ values are approximated with a deep neural network with weights θ. By using the transitions stored in a replay buffer, θ is learned by minimising the loss terms $(r_{k+1} + \gamma \max_{a'} Q_{\bar{\theta}}(s_{k+1}, a') - Q_\theta(s_k, a))^2$ with stochastic gradient descent, where $\bar{\theta}$ stands for the periodically updated copy of θ. This is the target network trick that DQN incorporates to battle the RL induced non-stationarity in the approximator learning. Another critical component is the replay buffer, that lets the agent save samples to be learned from over a long

course by also breaking the non-i.i.d. property of sequential collection. DQN's success has led it to be studied and enhanced further over years. The most prominent of these are summarised in Rainbow DQN [24]. Among these, we employ Dueling Networks [25] and Double Q-learning [26] in this paper.

B. Action Advising

Action advising [9] is a peer-to-peer knowledge exchange framework that requires only a common set of actions and a communication protocol between the peers. According to this, a learning peer (student) receives advice in the form of actions from a more knowledgeable peer (teacher) to accelerate its learning. These advice actions are generated directly from the teacher's decision-making policy; therefore, it is important for the student and the teacher to have the same goal in the task they are performing in. A key property that distinguishes this approach from the similar frameworks such as Policy Reuse [8] is the notion of budget constraint. By considering the realistic scenarios where it is not possible to reliably exchange information, the number of interactions in this framework is also limited with a budget. Consequently, the algorithms that operate in this problem setup should be capable of determining the most appropriate moments to exchange advice either from the perspective of teacher (teacher-initiated), student (student-initiated) or both (jointly-initiated).

IV. PROPOSED ALGORITHM

We adopt the MDP formalisation presented in Section III in our problem definition. The setup in this study includes an off-policy deep RL agent (student) with policy π_S learning to perform some task T in an environment with continuous state space and discrete actions. There is also another agent with policy π_T (teacher) that is competent in T. The teacher is isolated from the environment itself, but is reachable by the student via a communication channel for a limited number of times defined by the advising budget b. By using this mechanism, the student can request action advice $a_T = \pi_T(s)$ for its current state s. The objective of the student in this problem is to maximise its learning performance in T by timing these interactions to make the most efficient use of π_T.

Our approach provides a unified solution for addressing *when to ask for advice* and *how to leverage the advice* questions. In addition to the RL algorithm, the student is equipped with a neural network G_ω with weights ω that are not shared with the RL model in any way. The student also has a transitions buffer D with no capacity limit that holds the collected state-advice pairs. By using the samples in D, G_ω is trained periodically to provide the student an up-to-date imitation model of π_T to make it possible to reuse the previously provided advice. Moreover, G_ω is also used to determine what advice to collect by being regarded as a representation of D's contents. Obviously, making these decisions require G_ω to have a form of awareness of what it is trained on (in terms of samples). Therefore, G_ω employs Dropout regularisation in the fully-connected layers to have an estimation of epistemic uncertainty denoted by $G_\omega^\mu(s)$ for any state s as it is done in [21][13]. None of these components share

anything with or require access to the student's RL algorithm. This is especially advantageous when it comes to pairing up our approach with different RL methods.

At the beginning of the student's learning process, ω is initialised randomly and $D = \emptyset$. Then, at every timestep t in state s_t, the student goes through 3 stages of our algorithm: Collection, Imitation, Reuse. The remainder of this section describes these stages with the line number references to the complete flow of our algorithm summarised in Algorithm 1.

The collection stage (lines 13-19) remains active from the beginning until the student runs out of its advising budget b. At this step, the student attempts to collect advice if its current state has not been advised before. This is determined by the value of $G_\omega^\mu(s_t)$. If it is higher than the uncertainty threshold τ (which is set automatically in the imitation stage), it is decided that s_t is has not been advised before; thus, the student proceeds with requesting advice. However, if τ is undetermined, this request is carried out without performing any uncertainty check.

The imitation module is responsible for training G_ω and tuning τ accordingly. This stage (lines 20-22) is always active, but it is only triggered when these conditions that are checked at every timestep t are met: the student has collected n_{min} new samples in D (since the last imitation) or the student has taken t_{min} steps (since the last imitation) with at least $n_{min}/2$ new samples in D. Here, n_{min} and t_{min} are hyperparameters. These are set in order to keep the number of imitation processes within a reasonable number while also ensuring G_ω remains up-to-date with the collected advice. On one hand, if G_ω was updated for every new state-advice pair, it would be a very accurate model of D's contents, but the total training times would be a significant computational burden. On the other hand, if G_ω was updated infrequently, it would not cause any computational setbacks; however, it would not be a good representation of the collected advice either.

Once the imitation is triggered, G_ω is trained for k_{init} iterations (if it is the first ever training; else, for $k_{periodic}$ iterations) with the minibatches of samples drawn randomly from D. This process resembles the simplest form of behavioural cloning where the supervised negative log-likelihood loss $\mathcal{L}(\omega) = \sum_{(s,a)\in D} -log G_\omega(a \mid s)$ is minimised. Afterwards, τ is updated automatically to be compatible with the new state of this imitation network. This is done by measuring G_ω^μ and storing them in a set U for each s_i in D that satisfies $a_i = \text{argmax}_a G_\omega(a \mid s_i)$. Then, the uncertainty value that corresponds to the pth percentile (hyperparameter) in the ascending-order sorted U is assigned to τ. We do this to pick a threshold τ such that G_ω can consider these samples it classifies correctly as "known" while leaving a small portion that are likely to be outliers out, when G_ω^μ is compared with τ. This approach could be further developed by also considering the true-positive and false-positive rates, however, we opted for a simpler approach in this study.

Finally, the reuse stage (lines 23-26) handles the execution of the imitated advice whenever appropriate, to aid the student in efficient exploration. It becomes active as soon as the

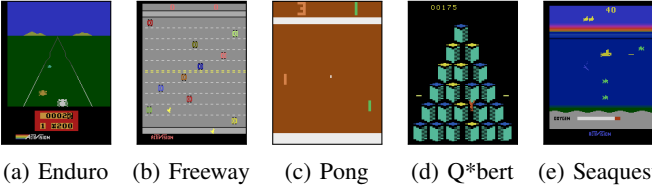(a) Enduro   (b) Freeway   (c) Pong   (d) Q*bert   (e) Seaquest

Fig. 1: Example RGB frames (observations) taken from the Arcade Learning Environment games Enduro (a), Freeway (b), Pong (c), Q*bert (d) and Seaquest (e).

imitation model G_ω is trained for the first time. Then, whenever $G_\omega^\mu(s_t) < \tau$ (i.e. G_ω is familiar with s_t), no advice collection is occurred at t and reusing is enabled for this particular episode, the student executes the imitated advice $\arg\max_a G_\omega(a \mid s_t)$. Unlike [13], we do not limit advice reusing to the exploration stage of learning, e.g. the period ϵ is annealed to its final value in ϵ-greedy. Instead, we define a reuse schedule that is independent than the underlying RL algorithm's exploration strategy. At the beginning of each episode, the agent either enables reuse module with a probability of ρ (set as ρ_{init} initially). This value is decayed until it reaches its final value ρ_{final} over t_ρ steps, similarly to ϵ-greedy annealing. This approach further eliminates the dependency of our algorithm to the RL algorithm's exploration strategy.

V. EXPERIMENTAL SETUP

We designed our experiments to answer the following questions about our proposal:

- How does our automatic threshold tuning perform against the manually-set ones in terms reuse accuracy and learning performance?
- How much does using advice imitation model to drive the advice collection process help with collecting more diverse state-advice dataset?
- Does collecting a dataset with more diverse samples make any significant impact on the learning performance?
- How much does every particular modification contribute in the final performance?

In the remainder of this section, we first describe our evaluation domain. Then, we provide the details our experimental process along with the substantial implementation details[1].

A. Evaluation Domain

In order to have the adequate amount of difficulty that is relevant to the modern deep RL algorithms, we chose the widely experimented Arcade Learning Environment (ALE) [27] that contains more than 60 Atari games as our testbed. We picked 5 well-known games among these, namely Enduro, Freeway, Pong, Q*bert, Seaquest, which involve different mechanics and present various learning challenges. In each of these games, the agent receives observations as RGB images with a size of $160 \times 210 \times 3$. These observations are converted

[1]The code for our experiments can be found at https://github.com/ercumentilhan/advice-imitation-reuse

---

Algorithm 1 Learning on a Budget via Teacher Imitation

1: **Input:** action advising budget b, student policy π_S, teacher policy π_T, number of training iterations t_{max}, initial reuse probability ρ_{init}, final reuse probability ρ_{final}, ρ decaying steps t_ρ, imitation network G_ω, number of imitation training iterations k_{init} and $k_{periodic}$, number of new samples and steps to trigger imitation n_{min} and t_{min}
2: $D \leftarrow \emptyset$ ▷ initialise empty state-advice buffer
3: *reuse_enabled* $\leftarrow False$ ▷ disable reuse by default
4: $\tau \leftarrow None$ ▷ no valid threshold
5: $\rho \leftarrow \rho_{init}$ ▷ set reuse probability with the initial value
6: $n_{last} \leftarrow 0$
7: **for** training steps $t \in \{1, 2, \dots t_{max}\}$ **do**
8:    **if** Env is reset **then**
9:      Set *reuse_enabled* $True$ with ϵ_{reuse} probability
10:      Get observation $s_t \sim Env$ if Env is reset
11:    **end if**
12:    $a_t \leftarrow None$ ▷ action is not determined yet
    ....
    ▷ Collection
13:    **if** *reuse_enabled* is $True$ **and** $b > 0$ **then**
14:      **if** G_ω is not trained **or** $G_\omega^\mu(s_t) > \tau$ **then**
15:        $a_t \leftarrow \pi_T(s_t)$ ▷ collect advice
16:        Add $\langle s_t, a_t \rangle$ to D
17:        $b \leftarrow b - 1$ ▷ decrease the budget
18:      **end if**
19:    **end if**
    ....
    ▷ Imitation
20:    **if** $|D| - n_{last} \geqslant n_{train}$ **or**
      $(|D| - n_{last} \geqslant n_{train}/2$ **and** $t - t_{train} \geqslant test)$ **then**
21:      Train G_ω with D for k_{init} or $k_{periodic}$ iterations
22:      $n_{last} \leftarrow |D|$
23:      Determine τ as described in Section IV
24:    **end if**
    ....
    ▷ Reuse
25:    **if** *reuse_enabled* is $True$ **and** a_t is $None$ **and**
      G_ω is trained **and** $G_\omega^\mu(s_t) < \tau$ **then**
26:      $a_t \leftarrow \arg\max_a G_\omega(a \mid s_t)$
27:    **end if**
28:    Decay ρ w.r.t. pre-defined schedule if $\rho > \rho_{final}$
    ....
29:    **if** a_t is $None$ **then**
30:      $a_t \leftarrow \pi_S(s_t)$ ▷ self policy
31:    **end if**
32:    Execute a_t and obtain $r_t, s_{t+1} \sim Env$
33:    Update the RL algorithm, e.g. DQN
34:    $s_t \leftarrow s_{t+1}$
35: **end for**

---

into $80 \times 80 \times 1$ grayscale images to reduce the amount of representational complexity, and are also stacked as the most recent 4 frames to eliminate the effect of partial observability.

Furthermore, since these games are originally processed at high frame-per-second rates with very little differences between two consecutive frames, agent actions are repeated for 4 frames by skipping frames. Consequently, the final $84 \times 84 \times 4$ sized observations the agent gets are built with the most recent 16 game frames. The range of rewards in these games are also different and unbounded. Therefore, to facilitate the stability of learning, they are clipped to be in $[-1, 1]$ before they are provided to the agent. The game episodes are limited to last for 108k frames (27k agent steps) at maximum.

B. Settings and Procedure

We experiment with an extensive set of agents to be able to determine the most beneficial enhancements included in our algorithm. The student agent variants we compare in our experiments are as follows:

- **No Advising (NA):** No form of action advising is employed, the agent relies on its RL algorithm only.
- **Early Advising (EA):** The student asks for advice greedily until its budget runs out. There is no further utilisation of advice beyond their execution at the time of collection. This is a simple yet well-performing heuristic.
- **Random Advising (RA):** The student asks for advice randomly with 0.5 probability. This heuristic uses the intuition that spacing out requests may yield more diverse and information rich advice.
- **EA + Advice Reuse (AR):** The agent employs the previously proposed advice reuse approach [13]. Advice is collected with early advising strategy, and the teacher is imitated with these advice. Then, advice are reused in place of the random exploration actions in approximately 0.5 of the episodes.
- **AR + Automatic Threshold Tuning (AR+A):** AR is combined with our automatic threshold tuning technique.
- **AR+A + Extended Reuse (AR+A+E):** AR is combined with both our automatic threshold tuning technique and the extended reusing scheme.
- **Advice Imitation & Reuse (AIR):** This agent mode incorporates all of our proposed enhancements (as detailed in Section IV). On top of AR+A+E, this mode also uses the imitation module's uncertainty to drive the advice collection process instead of relying on early advising.

We test the agents in learning sessions with length of 5M steps (equals to 20M game frames due to frame skipping) with an advising budget of 25k that corresponds to only 0.5% of the total number of steps in a session. At every $50k^{th}$ step, the agents are evaluated in a separate set of 10 episodes by having their action advising and exploration mechanisms disabled. The cumulative rewards obtained in these episodes are averaged and recorded as evaluation scores for that corresponding learning session step. This lets us measure the actual learning progress of the agents as the main performance metric.

The deep RL algorithm of the student agents is Double DQN with a neural network structure comprised of 3 convolutional layers (32 8×8 filters with a stride of 4 followed by 64 4×4 filters with a stride of 2 followed by 64 3×3 filters with a

TABLE I: Hyperparameters of the student's DQN and imitation module (for AR, AR+A, AR+A+E, AIR).

Hyperparameter name	Value
Discount factor γ	0.99
Learning rate	625×10^{-7}
Minibatch size	32
Replay memory min. size and capacity	50k, 500k
Target network update period	7500
ϵ initial, ϵ final, ϵ decay steps	1.0, 0.01, 500k
Learning rate	0.0001
Minibatch size	32

stride of 1) and fully-connected layers with a single hidden layer (512 units) and dueling stream output. For exploration, ϵ-greedy strategy with linearly decaying ϵ is adopted. The teacher agents are generated separately for each of the games prior to the experiments, by using the identical DQN algorithm and structure with the student. Even though the resulting agents are not necessarily at super-human levels achievable by DQN, they have competent policies that can achieve the evaluation scores of 1556, 28.8, 12, 3705, 8178 for Enduro, Freeway, Pong, Q*bert, Seaquest, respectively.

As we described in Section IV, our approach requires the student to be equipped with an additional behavioural cloning module that includes a neural network. We used the identical neural network structure to the student's DQN model except for the dueling streams. Fully-connected layers of this network are enhanced with Dropout regularisation with a dropout rate of 0.35 and the number of forward passes to measure the epistemic uncertainty via variance is set at 100.

The uncertainty threshold for AR is set as 0.01 for every game. Determining a reasonable value for this parameter requires to access the tasks briefly, which we have performed prior to the experiments; even though this will not be reflected at the numerical results, it should be noted that this is a critical disadvantage of AR. The automatic threshold tuning percentile used in AR+A, AR+A+E, AIR is set as 90. This is a very straightforward hyperparameter to adjust compared to the (manual) uncertainty threshold itself and can potentially be valid in a wide variety of tasks. For the extensive reuse scheme in AR+A+E and AIR, we set ρ_{init} and ρ_{final} as 0.5 and 0.1, respectively. We defined the annealing schedule to begin at $500k^{th}$ step and last until $2M^{th}$ step. For the imitation triggering conditions in AIR, n_{min} is set as 2.5k (samples) and t_{min} is set as 50k (timesteps). Finally, the number of imitation network training iterations is set as 200k for the initial one (applies to all modes but NA, EA and RA) and 50k for the periodic ones (only applies to AIR).

All of the aforementioned hyperparameters reported in this section are set empirically prior to the experiments and are kept the same across every game. The most significant ones among the unmentioned hyperparameters of the student's learning components are presented in Table I.
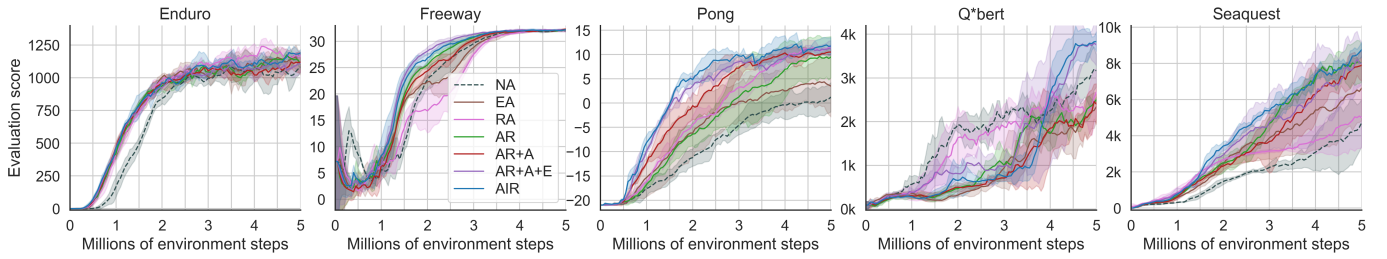
Fig. 2: Evaluation scores obtained by the agent modes NA, EA, RA, AR, AR+A, AR+A+E, AIR in the ALE games Enduro, Freeway, Pong, Q*bert, Seaquest. Shaded areas show the standard deviation across 3 runs.
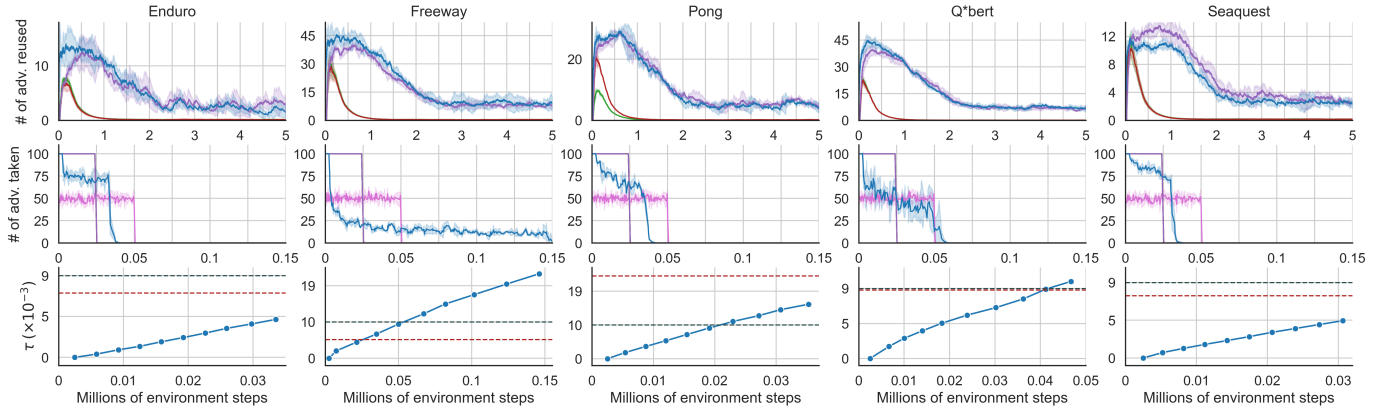

Fig. 3: Top and middle rows show the plots for the number of advice reuses and advice collections (for the first 5M and 150k steps, respectively) in every 100 steps performed by the relevant student modes. The legend for colours is the same as Figure 2 and the shaded areas show the standard deviation across 3 different runs. Bottom row shows the values of τ across a single learning session, with different timestep scopes determined by the length of AIR's advice collection stage. Blue colour represents AIR, the dashed grey lines represent AR, and the dashed red lines represent AR+A and AR+A+E.
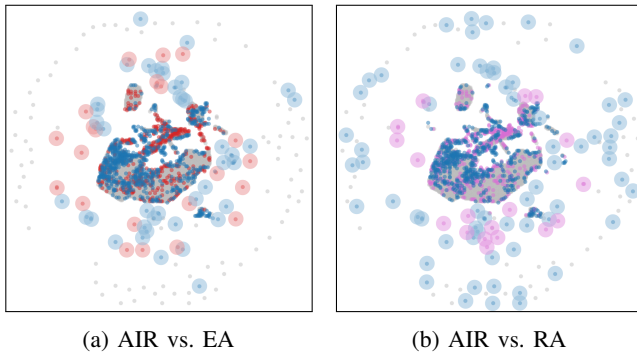

(a) AIR vs. EA   (b) AIR vs. RA

Fig. 4: UMAP [28] embeddings of the advice collected states in Seaquest by AIR (blue) vs. EA (red) in (a) and AIR (blue) vs. RA (pink) in (b), where the samples in common are shown in grey. Areas denoted with larger circles are the outliers covered only by either AIR (blue), EA (red) or RA (pink).

VI. RESULTS

The results of our experiments in Enduro, Freeway, Pong, Q*bert and Seaquest are presented in several plots to let us analyse the performance of the student modes NA, EA, RA, AR, AR+A, AR+A+E, AIR in different aspects. Figure 2 contains

the evaluation scores plots. In Figure 3, plots for the number of advice reuses per 100 steps (top row) performed by AR, AR+A, AR+A+E, AIR; plots for the number of advice collections per 100 steps performed by NA, RA, AR, AR+A, AR+A+E, AIR (middle row); plots for the values of the τ hyperparameter (uncertainty threshold) used by AR, AR+A, AR+A+E, AIR (bottom row) in a single run are shown. The shaded areas in these plots show the standard deviation across 3 runs. The final evaluation scores, percentage of advice reuses in total number of environment steps as well as their accuracies are presented in Table II. Finally, in order to highlight the differences in the advice-collected state diversity of EA (identical collection strategy to AR, AR+A, AR+A+E), RA and AIR, these states from a single Pong run are visualised with the aid of UMAP [28] dimensionality reduction technique in Figure 4. Here, the scatter plot on the left compares AIR (blue) vs. EA (red) and the one on the right compares AIR (blue) vs. RA (pink); and the samples collected in common are shown in grey.

We first analyse the learning performances via evaluation scores. In Enduro, all methods but NA have a very similar learning speed and final scores, with AIR being slightly ahead of the rest. In Freeway, they all achieve nearly the same final scores, but they are distinguishable with small differences in

TABLE II: Final evaluation scores, percentages of advice reuses in the total environment steps, percentages of advice reuse accuracies achieved by NA, EA, RA, AR, AR+A, AR+A+E, AIR (+ signs are omitted) in 5 ALE games aggregated over 3 runs. The standard deviations across runs are indicated with ±. The best scores/accuracies are denoted in bold.

Game	Mode	Evaluation Score Final	Advice Reuse Ratio (%)	Accuracy (%)
Enduro	NA	1066.34 ± 37.3	—	—
	EA	1131.13 ± 69.0	—	—
	RA	1170.60 ± 19.0	—	—
	AR	1127.72 ± 26.9	0.49 ± 0.03	70.06 ± 0.6
	ARA	1117.93 ± 59.0	0.45 ± 0.04	72.36 ± 0.3
	ARAE	1102.44 ± 62.4	4.97 ± 0.44	$\mathbf{75.24 \pm 0.5}$
	AIR	$\mathbf{1184.02 \pm 19.6}$	4.79 ± 0.44	73.63 ± 0.2
Freeway	NA	31.98 ± 0.1	—	—
	EA	32.09 ± 0.1	—	—
	RA	32.02 ± 0.2	—	—
	AR	32.06 ± 0.1	1.60 ± 0.11	93.02 ± 0.3
	ARA	$\mathbf{32.26 \pm 0.1}$	1.53 ± 0.10	93.48 ± 0.1
	ARAE	32.14 ± 0.0	15.76 ± 0.08	$\mathbf{95.43 \pm 0.1}$
	AIR	32.14 ± 0.1	17.89 ± 0.47	95.39 ± 0.2
Pong	NA	0.95 ± 2.4	—	—
	EA	3.73 ± 4.9	—	—
	RA	11.48 ± 0.2	—	—
	AR	9.41 ± 3.5	0.52 ± 0.02	79.80 ± 0.5
	ARA	10.48 ± 0.4	0.93 ± 0.01	75.07 ± 0.7
	ARAE	11.21 ± 1.2	10.4 ± 0.59	79.00 ± 0.1
	AIR	$\mathbf{11.81 \pm 1.2}$	9.98 ± 0.36	$\mathbf{80.46 \pm 0.8}$
Q*bert	NA	3154.91 ± 408.9	—	—
	EA	2277.98 ± 300.5	—	—
	RA	2528.70 ± 505.4	—	—
	AR	2434.70 ± 54.8	0.93 ± 0.04	80.37 ± 0.6
	ARA	2359.39 ± 371.9	0.93 ± 0.02	78.86 ± 0.7
	ARAE	3763.72 ± 340.3	14.57 ± 0.25	$\mathbf{92.87 \pm 0.7}$
	AIR	$\mathbf{3814.34 \pm 134.6}$	15.16 ± 0.21	92.83 ± 0.2
Seaquest	NA	4496.41 ± 1101.0	—	—
	EA	6538.18 ± 1445.1	—	—
	RA	5033.04 ± 1413.3	—	—
	AR	8053.93 ± 935.9	0.61 ± 0.03	72.96 ± 0.8
	ARA	7851.03 ± 556.7	0.56 ± 0.03	$\mathbf{76.29 \pm 0.7}$
	ARAE	8082.69 ± 1105.2	5.93 ± 0.49	74.18 ± 1.5
	AIR	$\mathbf{8614.04 \pm 268.7}$	4.79 ± 0.20	74.87 ± 0.5

learning speed where AR+A+E is on the top followed by other advanced student modes AIR, AR, AR+A. When we move to Pong, Q*bert and Seaquest, we finally see the student mode performances to be more distinctive. Even though the basic heuristics (RA and EA) show that a little number of advice from a competent policy can make substantial boosts on learning, these modes fall behind of those that employ advice reuse and fail to be a reliable choice, i.e. performing worse than NA in Freeway and Q*bert. Overall, the best mode AIR and the runner-up AR+A+E are ahead of all, with AR and AR+A following them.

Among the advice reuse approaches, we see that the most beneficial modification is the extended reuse schedule (+E) as it is highlighted by the difference between AR+A and AR+A+E. Defining such a schedule independently from the student's RL exploration strategy involves using some extra hyperparameters, nevertheless, they are rather trivial to set arbitrarily. The trends

in the advice reuse plots (Figure 3, top row) show how these schedules differ. The versions with +E (AIR and AR+A+E) yield around $10\times$ more reusing, which apparently plays an important role in the performance improvement. However, it is still not clear how to define the optimal reuse schedules.

Automatic threshold tuning (+A) also performs comparably, if not better, with the manual tuning approach (AR) as we can observe in the evaluation scores. Additionally, AR+A managed to achieve very similar reuse accuracies with AR; this also supports its success. When we examine the τ values, we see AR+A determined values that are close to the hand-tuned ones, except for the case in Pong where the difference is more significant. This reflected in reuse trend and evaluation performance, giving AR+A a very advantageous head start. These results support the idea that +A a far more preferable approach considering how problematic it can be to tune the sensitive τ threshold manually. For instance, if they were to deployed in some significantly different domains as they are, then we could potentially see AR+A coming far ahead of AR with a poorly tuned τ. Furthermore, the periodic imitation model updates incorporated in AIR makes +A an essential component. In the bottom row of Figure 3, we also show how AIR changes its τ values over time as it collects more advice samples and updates its imitation model accordingly. Clearly, it is very difficult to manage these changes manually.

Finally, we also see that collecting advice by utilising the imitation model's uncertainty (as it is done by AIR) contributes to the agent's learning. When we look at the advice collection plots, we see 3 different types of behaviours: early collection (EA, AR, AR+A, AR+A+E), random collection (RA), and AIR. Even though they seem to be occurring mostly in the same time windows, AIR does this in an uncertainty-aware fashion; hence the decreasing collection rate over time. Freeway is the case in which AIR is very selective. This is possibly due to the fact that in Freeway, the agent can traverse only a limited space which consequently reduces the diversity of the acquired observations. Nonetheless, this is not reflected in the evaluation scores as dramatically due to this game being rather trivial to solve. Another interesting observation is made by analysing the advice collected states in Seaquest by AIR, EA (which is identical to AR, AR+A, AR+A+E in terms of collection strategy), RA in a reduced dimensionality as seen in Figure 4. We chose Seaquest since it is the game where AIR is significantly ahead of AR+A+E, which can be credited to AIR's only difference from it (collection strategy). We also include RA here mainly because it can potentially do better in acquiring different samples than EA. Here, the large circles denote the outliers (diverse samples) that are only covered only by either AIR, EA or RA. These are the important bits to pay attention to and compare. As it can be seen, AIR yields larger coverage, i.e. more diverse dataset of advice, in both cases against EA and RA collection strategies.

VII. CONCLUSIONS AND FUTURE WORK

In this study, we proposed an automatic threshold tuning technique, an extended advice reusing schedule and an imita-

tion model uncertainty-based advice collection procedure by extending the previously proposed advice reusing algorithm. We also developed a combined approach by incorporating these components, that is able to collect a diverse set of advice to build a more widely applicable advice imitation model for advice reuse.

The experiments in 5 different Atari games from the ALE domain have shown that our enhancements provide significant improvements over the baseline advice reuse method as well as the basic action advising heuristics. First, being able to tune the uncertainty thresholds on-the-fly was observed to yield the learning performance of the carefully tuned threshold, which require unrealistic access to the tasks and extra effort to be adjusted. Secondly, we found that having the advice reusing process span across a larger portion of the learning session rather than just the steps that involve random exploration can yield superior performance. However, defining the best schedule for the maximum advice utilisation efficiency remains to be an open question. Thirdly, the uncertainty-driven advice collection method was found to be successful way to improve the imitation module's dataset diversity. Nevertheless, periodic training process can be improved with better incremental learning techniques to make a better use of this simultaneous collection-imitation idea. Finally, our unified algorithm demonstrated state-of-the-art performance across 5 Atari games by performing either on-par or better than its closest competitors.

The future extensions of this work can involve experimenting with more principled Policy Reuse approaches in the literature to further improve the advice reuse strategy. Furthermore, it will be a worthwhile study to make the teacher imitation better at learning online from the new samples it acquires. Finally, even though it is in the core motivation of our approach not to access and modify the agent's RL components, it will be beneficial to investigate Learning from Demonstrations techniques and their possible contributions in our framework.

REFERENCES

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] J. Schrittwieser, I. Antonoglou, T. Hubert, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *CoRR*, vol. abs/1911.08265, 2019.

[3] OpenAI, I. Akkaya, M. Andrychowicz, *et al.*, "Solving rubik's cube with a robot hand," *CoRR*, vol. abs/1910.07113, 2019.

[4] A. A. Taïga, W. Fedus, M. C. Machado, A. C. Courville, and M. G. Bellemare, "Benchmarking bonus-based exploration methods on the arcade learning environment," *CoRR*, vol. abs/1908.02388, 2019.

[5] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Com.*, vol. 3, no. 1, pp. 88–97, 1991.

[6] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, MIT Press, 1996, pp. 1040–1046.

[7] T. Hester, M. Vecerík, O. Pietquin, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, AAAI Press, 2018, pp. 3223–3230.

[8] F. Fernández and M. M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, ACM, 2006, pp. 720–727.

[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] 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.

[11] 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*, AAAI Press, 2020, pp. 5792–5799.

[12] E. Ilhan, J. Gow, and D. Perez-Liebana, *Student-initiated action advising via advice novelty*, 2021. arXiv: 2010.00381.

[13] E. Ilhan, J. Gow, and D. Perez-Liebana, "Action advising with advice imitation in deep reinforcement learning," in *Proceedings of the 20th Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2021, May 3-7, 2021*, IFAAMAS, 2021.

[14] M. E. Taylor, N. Carboni, A. Fachantidis, I. P. Vlahavas, and L. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connect. Sci.*, vol. 26, no. 1, pp. 45–63, 2014.

[15] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-Student Framework: A Reinforcement Learning Approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[16] 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*, IJCAI/AAAI Press, 2016, pp. 804–811.

[17] 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*, ACM, 2017, pp. 1100–1108.

[18] C. Zhu, Y. Cai, H.-f. Leung, and S. Hu, "Learning by reusing previous advice in teacher-student paradigm," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*, International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1674–1682.

[19] S.-A. Chen, V. Tangkaratt, H.-T. Lin, and M. Sugiyama, "Active deep q-learning with demonstration," *CoRR*, vol. abs/1812.02632, 2018.

[20] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by random network distillation," *CoRR*, vol. abs/1810.12894, 2018.

[21] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, vol. 48, 2016, pp. 1050–1059.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[24] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, AAAI Press, 2018, pp. 3215–3222.

[25] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, ser. JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 1995–2003.

[26] 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*, AAAI Press, 2016, pp. 2094–2100.

[27] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.

[28] L. McInnes and J. Healy, "UMAP: uniform manifold approximation and projection for dimension reduction," *CoRR*, vol. abs/1802.03426, 2018.