Action Advising with Advice Imitation in Deep Reinforcement Learning

Ercüment İlhan
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
e.ilhan@qmul.ac.uk

Jeremy Gow
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
jeremy.gow@qmul.ac.uk

Diego Perez-Liebana
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
diego.perez@qmul.ac.uk

ABSTRACT

Action advising is a peer-to-peer knowledge exchange technique built on the teacher-student paradigm to alleviate the sample inefficiency problem in deep reinforcement learning. Recently proposed student-initiated approaches have obtained promising results. However, due to being in the early stages of development, these also have some substantial shortcomings. One of the abilities that are absent in the current methods is further utilising advice by reusing, which is especially crucial in the practical settings considering the budget constraints in peer-to-peer interactions. In this study, we present an approach to enable the student agent to imitate previously acquired advice to reuse them directly in its exploration policy, without any interventions in the learning mechanism itself. In particular, we employ a behavioural cloning module to imitate the teacher policy and use dropout regularisation to have a notion of epistemic uncertainty to keep track of which state-advice pairs are actually collected. As the results of experiments we conducted in three Atari games show, advice reusing via imitation is indeed a feasible option in deep RL and our approach can successfully achieve this while significantly improving the learning performance, even when it is paired with a simple early advising heuristic.

KEYWORDS

Deep Reinforcement Learning; Deep Q-Networks; Action Advising

ACM Reference Format:
Ercüment İlhan, Jeremy Gow, and Diego Perez-Liebana. 2021. Action Advising with Advice Imitation in Deep Reinforcement Learning. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep reinforcement learning (RL) has made it possible to build end-to-end learning agents without having to handcraft task-specific features, as it is showcased in various challenging domains such as StarCraft II [29] and DotA II [3] in the recent years. These feats make deep RL a great candidate to be employed in complex real-world sequential decision-making problems. However, achieving the reported levels of performance usually requires millions of environment interactions due to the deep learning induced complexity as well as the exploration challenges in RL itself. Even though this may seem negligible in most of the experimental domains considering the immense amount of computing power available to be utilised through parallel simulations, it usually poses a problem in the real-world scenarios due to the interaction costs and safety concerns. Furthermore, since RL is an inherently online learning approach, it is desired for the agents to be continually learning after they have been deployed too. For these reasons, it is crucial to improve sample efficiency in deep RL, which is actively investigated in several lines of research. One promising approach to tackle this setback is leveraging some legacy knowledge acquired from other entities such as agents, programs or humans.

Peer-to-peer knowledge transfer in deep RL has been investigated in various forms to this date [9]. A popular approach, namely Learning from Demonstrations (LfD), focuses on incorporating a previously recorded dataset in the learning process. By taking some dataset generated by another competent [15] or imperfect [13] peer, the learning agent tries to make the most out of the available information through off-policy learning and extra loss terms. Another promising, yet under-investigated class of techniques, namely Action Advising [27], aims to take advantage of a competent peer interactively when there is no pre-recorded data. The learning agent acquires advice in the form of actions from a teacher for a limited number of times defined by a budget that resembles the practical limitations of communication and attention. This approach is especially beneficial in the situations where there is no way to access the actual task before the online training, data collection is costly or the relevant data that will do the most contribution in the learning can not be determined. Action advising methods in deep RL today are quite limited and therefore have several shortcomings. An important one of these as we address in this study is not being able to make further use of the advice beyond its collection.

The scope of the action advising problem is generally limited to answering "*when* to ask for advice?". It is commonly not of any interest how the collected advice is utilised by the student agent's task-level RL algorithm, e.g., how it is stored, replayed, discarded; especially since these are dealt with by the studies that focus on off-policy experience replay dynamics in general [10, 23], or the specific case of having demonstration data as in LfD. However, even without interfering with the student's task-level learning mechanism, it is still possible to make more of advice through reuse. Current action advising algorithms in deep RL have no way of telling if they have asked for advice in a very similar or even identical state already in the learning session. Thus, they do not record these in any way, and usually end up requesting advice from the teacher redundantly. In order to address this, we incorporate a separate neural network

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

to do behavioural cloning [22] on the samples (state-action pairs which are equal to the state-advice pairs in the context of action advising) collected from the teacher. This network then will be able to serve as a state-conditional generative model that will let us sample advice for any given observation. However, since this model should also have a notion of distinguishing the recorded states from the unrecorded ones to avoid producing false advice for unfamiliar states, we also propose incorporating a well known regularisation mechanism called Dropout [24] within this network to serve as an epistemic uncertainty estimator [12] which will allow the student to determine whether the state is recorded by comparing this estimation with a threshold.

Our contributions in this study are as follows: First, we show that it is possible to generalise teacher advice across similar states in deep RL with high accuracy. Second, we present a RL algorithm-agnostic approach to memorise and imitate the collected advice that is suitable with the deep RL settings. Finally, we demonstrate that advice reuse via imitation provides significant boosts in the learning performance in deep RL even when it is paired with a simple baseline like early advising.

2 RELATED WORK

The majority of action advising studies to date have been conducted in classical RL settings. [27] was the first study to formalise action advising within a budget constrained teacher-student framework. Specifically, they studied the teacher-initiated scenario and came up with several heuristics to distribute the advising budget to maximise the student's learning performance, such as early advising and importance advising. This work was then extended to introduce several new state importance metrics [26]. In [33], action advising problem was approached as a meta-RL problem itself. Instead of relying on heuristics, the authors attempted to learn the optimal way to distribute advising budget by using a measurement of the student's learning acceleration as the meta-level reward. Besides these studies that only consider the teacher's perspective, [1] explored student-initiated and jointly-initiated variants considering the impracticality of requiring the teacher's attention constantly. They achieved results on-par with the previous work without requiring the teacher full-time. [31] shed light in the theoretical aspects of action advising problem by using a more general setting involving multiple teachers and demonstrated the effects of having good or bad teachers. In [7], the authors adopted the teacher-student framework in cooperative multi-agent RL where the agents learn from scratch and hold no assumptions of their teacher roles and expertise. By proposing state counting as a new heuristic in this setting, they successfully accelerate team-wide learning of independent learners. More recently, learning to teach concepts was further investigated in [11] with a focus on the properties that make for a good teacher. In this work, besides learning *when* to advise, the teachers also learn *what* to advise. Similarly, [21] adopted the meta RL approach, this time as a deep RL scale. They considered a team of two agents that learn to cooperate from scratch in tabular multi-agent tasks. [32] is one of the most recent studies conducted in tabular settings. The idea of reusing the previously collected advice in order to make the most out of a given small budget was studied. By devising several heuristics to serve as reusing schedules, they demonstrated promising results that outperform the algorithms incapable of advice reusing.

The domain of deep RL is a fairly new area for action advising where the primary choice is the student-initiated approaches. [6] is one of the first studies to explore the idea of action advising in deep RL. They combined the LfD paradigm [15] with interactive advice exchange under the name of *active learning from demonstrations* to collect demonstration data on-the-fly to be utilised via imitation capable loss terms as used in [15]. Furthermore, they proposed using epistemic uncertainty estimations of the student agent's model to time this advice. Later, [18] was proposed as an extension of [21]. This time, meta deep RL to address learning to teach idea was applied in the problems that are deep RL in the task-level. Through multiple centralised learning sessions, agents in a set of cooperative multi-agent tasks were made to learn taking student and teacher roles as needed in order to improve team-wide knowledge. To do so, they adopted *hierarchical reinforcement learning* [20] to deal with the meta-level credit assignment problem of the teacher actions. In [16], the formal action advising framework was scaled up to deep RL level for the first time. Similarly to [7], a team of agents in a cooperative multi-agent scenario were made to exchange advice by embracing teacher or student roles as needed. This was accomplished by using *random network distillation* (RND) [4] to replace state counting with state novelty, hence introducing a new heuristic that is applicable in non-linear function approximation domain. Later on, [8] proposed the idea of uncertainty-based action advising as in [6], though without employing any additional loss terms. To access uncertainty estimations, they studied the case of student agent with a multi-headed network architecture in particular. In a more recent work [17], student-initiated scenario is further studied to devise a more robust heuristic able to handle extended periods of absence of teacher as well as having no requirements in the student's task-level architecture by completely decoupling the module that is responsible for advice timing from the student's model. Even though this method also uses the state novelty heuristic proposed in [16], they operated on the advised states directly rather than every encountered state.

Clearly, none of the related work in deep RL addressed further utilisation of collected advice, besides [6] which does it through interfering with the student's learning mechanism (via a custom loss function), unlike our approach. The study that is closest to the idea we present in this paper is [32]; though, it is limited to the tabular RL domains only. Such a setting makes it more straightforward for the agent to precisely memorise the state-advice pairs in a look-up table to be able to reuse anytime. Furthermore, the executed advice usually has an instantaneous impact on the agent behaviour in the case of tabular RL, which presents unique options to assess their usefulness. Since these advantages are absent in deep RL, our work deals with different challenges than those in [32].

3 BACKGROUND

3.1 Reinforcement Learning

Reinforcement Learning (RL) [25] is a trial-and-error learning paradigm that deals with sequential decision-making problems where the environment dynamics are unknown. In RL, Markov Decision Process (MDP) formalisation is used to model the environment

and the interactions within. According to this, an environment is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ where $\mathcal{S}$ is the finite set of states, $\mathcal{A}$ is the finite set of actions, $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ defines the state transitions and $\gamma \in [0, 1]$ is the discount factor. The agent to interact within an environment receives a state observation s_t at each timestep t, and executes an action a_t to advance to the next state s_{t+1} while obtaining a reward r_t. Actions of the agent are determined by its policy $\pi: S \rightarrow A$, and the agent's objective is to construct a policy that maximises the expected sum of discounted rewards in any timestep, which can be formulated as $\sum_{k=0}^{T} \gamma^k r_{t+k}$ for a horizon of T timesteps.

3.2 Deep Q-Networks

Deep Q-Network (DQN) [19] is a prominent RL algorithm that tries to obtain the optimal policy in complex domains by employing non-linear function approximation via neural networks to learn mapping any given state into state-action values ($Q(s, a)$). Specifically, a neural network G_θ with randomly initialised weights θ is trained over the course of learning to minimise the loss $(r_{k+1} + \gamma \max_{a'} Q_{\bar{\theta}}(s_{k+1}, a') - Q_\theta(s_k, a))^2$ with batches of transitions that are collected on-the-fly and stored in a component called replay memory. Periodically using the samples from this memory, which is referred to as *experience replay*, is an essential mechanism in DQNs. As well as improving sample efficiency by reusing samples multiple times, it also breaks the non-i.i.d. property of sequentially collected data. Furthermore, DQNs also employ another trick to aid convergence. Since both the Q-value targets and network weights are learned at the same time, there is a significant amount of non-stationarity seen in these target values used in the loss function, which may introduce further instabilities due to the bootstrapped updates. In order to alleviate this, a separate copy of G is held with weights $\bar{\theta}$ that are updated periodically with copies of θ, to be used in the target term in the loss function.

Due to its end-to-end learning and discarding the need for hand-crafted features, DQN has become a very popular approach in the field of RL that is followed by further enhancements over the years. The most substantial ones among these are identified and combined in a version called Rainbow DQN [14]. In our study, we employ double Q-learning [28] and dueling networks [30] among these essential modifications.

3.3 Behavioural Cloning

Behavioural cloning [22] refers to the ability of imitating a demonstrated behaviour. It is especially useful in the situations where it is more difficult to specify reward functions than to provide some expert demonstration. The simplest way of achieving this in the domain of deep RL is to train a non-linear function approximator, e.g. neural network G_ω with weights ω, through supervised learning on the provided demonstration samples in the form of state-action pairs denoted by $\langle s, a \rangle$. This is done by treating these as i.i.d. samples and minimising an appropriate loss function such as $\mathcal{L}(\omega) = \sum_{(s,a) \in D} -log G_\omega(a \mid s)$. Consequently, a state-conditional generative model is obtained that is capable of imitating the expert actions for the demonstrated states. In practice, however, this approach is unreliable to be used as a task policy as it is. This is because the agent often encounters states that are not contained in the provided dataset, and therefore, end up exhibiting sub-optimal behaviour in these states which lead to further divergence in the trajectories. However, adopting the idea in this most basic form is sufficient in our study as it provides us the adequate functionality of generating actions correctly for the states we ensure G_ω is trained with.

3.4 Dropout

Dropout [24] is a simple yet powerful regularisation method developed to prevent neural networks from overfitting. Its working principle is based on involving some random noise in the hidden layers of the networks. A neural network layer with the feed-forward operation can be described as $\boldsymbol{y} = f(\boldsymbol{w}\boldsymbol{x} + b)$, where the output is $\boldsymbol{y} \in \mathbb{R}^q$, the input is $\boldsymbol{x} \in \mathbb{R}^p$, the network weights for this particular layer are $\boldsymbol{w} \in \mathbb{R}^{q \times p}$ and $b \in \mathbb{R}^q$, f is any activation function, for input size of p and output size of q. In a layer with dropout, this equation takes the form of $\boldsymbol{y} = f(\boldsymbol{w}\tilde{\boldsymbol{x}} + b)$ where $\tilde{\boldsymbol{x}} = \boldsymbol{r} * \boldsymbol{x}$ represent randomly dropped out input which is determined by $r \sim Bernoulli(p)$. Hence, the learning process gets to be regularised with this random noise which is re-determined in every forward pass. The value p controls the rate of dropout and is responsible for the regularisation strength.

In addition to its regularisation capability, dropout can also be used to estimate epistemic uncertainty of a neural network model, as shown in [12]. For any particular input, performing forward passes multiple times yield different outputs due to the dropout induced stochasticity, which can be treated as an approximation of probabilistic deep Gaussian process. Following this idea, the variance in these output values can therefore be interpreted as a representation of the model's uncertainty. Finally, since these forward passes can be performed concurrently, this approach provides a practically viable option to evaluate the uncertainty in deep learning models.

3.5 Action Advising

Action advising [27] is a knowledge exchange approach built on the teacher-student paradigm. Requiring only a common set of actions and a communication protocol between the teacher and the student makes this a very flexible framework. In its originally proposed form, the learning agent (student) is observed by an experienced peer (teacher) and is given action advice to be treated as high quality explorative actions to accelerate its learning. However, maximum number of these interactions are limited with a budget constraint considering the real-world conditions where communication and attention span are usually limited. Therefore, the approaches that adopt this idea address the question of *when* to exchange advice in order to maximise the learning performance. This is usually accomplished either by performing meta-learning over multiple learning sessions or by following heuristics as we do in this study.

Currently, there are several heuristic approaches with varying complexities and advantages in the deep RL domain such as early advising, random advising, uncertainty-based advising and novelty-based advising. In this paper, we incorporate early advising as the baseline to build our method on. Despite its simplicity, this method performs very well in deep RL especially with small budget scenarios [17]. This is because the earlier samples have far more impact

on the learning in deep RL models since providing high quality transitions that contains rewards provide more stable Q-value targets early on which can significantly reduce the non-stationarity in the learning process. Finally, since the teacher is followed consistently in this approach, the student is more likely to encounter the critical states that would require deep exploration. This is an important property to have when it comes to spending the budget wisely.

4 PROPOSED APPROACH

We follow the standard MDP formalisation given in Section 3.1 in our problem definition. In this setting, a student agent that employs an off-policy deep RL algorithm performs learning in an episodic single-agent environment through trial-and-error interactions. It receives an observation s_t and then executes an action a_t generated by its policy π_S to receive a reward r_t at each timestep t, in order to maximise its cumulative discounted rewards in any episode. According to the teacher-student paradigm (Section 3.5) we adopt, there is also an isolated peer that is competent in this same task, and is referred to as the teacher. For a limited number of times defined by the action advising budget b, the student is allowed to acquire an action advice from the teacher for the particular state s it is in. While the teacher can have its own teaching strategies to generate actions to advise, in our setting, we determine the action to be advised greedily from the teacher's behaviour policy as $\pi_T(s)$. This is a commonly followed approach with the assumption of the teacher and the student's optimal task-level strategies are equivalent. The student considers this advice as a part of a high-reward strategy and follows them upon collection. In this final form of the problem, the student's objective is to spend its budget at the most appropriate times to maximise its learning performance.

We aim to devise a method that will enable the student to memorise the collected advice to be able to re-execute them in the similar states; therefore, avoiding wasting its budget in redundant states and potentially being able to follow the teacher advice many more times than its budget. In tabular RL, this is trivial to achieve simply by storing the advised actions paired with the states in a look-up table. When it comes to deep RL where any particular observation is not expected to be encountered more than once, however, there needs to be a generalisable approach. For this purpose, we propose the student agent to employ a separate behavioural cloning module, which consists of a neural network as the state-conditional generative model $G_\omega : \mathcal{S} \rightarrow \mathcal{A}$. By training G_ω in a supervised fashion with the obtained state-advice pairs (stored in a buffer D) to minimise the negative log-likelihood loss $\mathcal{L}(\omega) = \sum_{(s,a)\in D} -logG_\omega(a \mid s)$, the student can imitate the teacher's advice to reuse them accordingly. However, this method does not have any mechanisms to prevent the student from generating incorrect advice from the states it has not collected. Therefore, we also employ Dropout regularisation in G_ω in order to grant this behavioural cloning module a notion of epistemic uncertainty through measuring the variance in the outputs obtained from multiple forward passes for a particular input state. We denote this uncertainty estimation by $G^{\mu}_{\omega}(s)$. The states G_ω is trained on will be less susceptible to the variance caused by the dropout and yield smaller uncertainty values. By this means, the student can determine how likely a state is to be already recorded as advised when it comes to reusing them, and can make a decision according to a threshold.

An obvious question regarding the feasibility of reusing advice in deep RL arises here: can the teacher's advice be generalised over similar states accurately? As we investigate in the experiments in Section 7, actions generated by the teacher policy usually span over similar states. Clearly, the uncertainty threshold to consider a state as recorded is responsible for the trade-off between the reusing amount and the accuracy of the self-generated teacher advice. A small threshold value makes the student reuse its budget in fewer states with higher accuracy, whereas a larger value results in more frequent reusing with lower accuracy.

The detailed breakdown of our approach is summarised with an emphasis on the proposed modifications as follows (as also shown in Algorithm 1): The student starts with a randomly initialised G_ω and empty D. At each timestep t with the (observed) state s_t and an undecided action a_t, the student first checks if D has any new samples. As soon as D reaches the size defined by n_D, G_ω is trained with mini-batch gradient descent over the samples in D for k_{bc} iterations. Afterwards, if the environment was reset (a new episode started), the student determines whether to enable advice reuse via imitation for this particular episode with a probability of ε_{reuse}, which is combined with other conditions too later on in the algorithm. The idea behind employing this condition is to ensure that the student can also execute its own exploration policy in order to increase the data diversity in its replay memory, which is crucial to improve the quality of learning. Furthermore, determining this variable on an episodic basis lets the agent follow consistent policies in the exploration steps, rather than dithering between two policies. In the next phase, the student deals with the advice collection. We adopt the simple yet strong baseline of early advising here. According to this, the agent just collects advice without any conditions until its budget runs out. In the next phase, the student decides whether to reuse advice generated by its G_ω. There are several conditions to be satisfied for this to occur in addition to the advice reuse being allowed for this particular episode. Firstly, a_t must be non-determined, which implies the agent has not collected any advice from the teacher already. Secondly, G_ω must be already trained, so that it can generate meaningful actions. Then, the student also checks if its own action $\pi_S(a \mid s_t)$ is explorative. This condition limits the action advising actions to the exploration steps only in order to prevent overriding the student's actual policy which may result in lack of Q-value corrections and cause deteriorative effects when too much advising occurs. Finally, it is checked whether $G^{\mu}_{\omega}(s_t)$ is smaller than the reuse threshold τ_{reuse}. Incorporating such threshold is important to limit the imitated advice to the states that have low uncertainty according to G_ω to achieve higher accuracy of generating correct teacher actions. On one hand, having this threshold too high would make the student consistently follow G_ω which would result in a dataset with lower diversity. On the other hand, if τ_{reuse} is set too small, then G_ω would be ignored in the most of the cases and the student would be following its own exploration policy. After all these steps, if a_t is still non-determined, the student follows its own policy and decide a_t by $\pi_S(a \mid s_t)$.

Algorithm 1 Action Advising with Advice Imitation

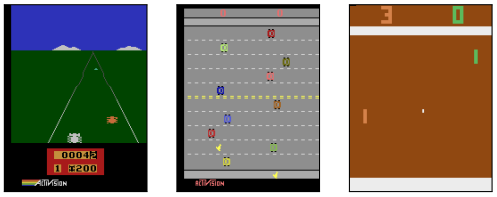
Input: action advising budget b, student policy π_S, teacher policy π_T, number of training iterations t_{max}, advice reuse uncertainty threshold τ_{reuse}, advice reuse probability (episodic) ε_{reuse}, behavioural cloning variables:
- generative network G_ω (G^μ_ω denotes uncertainty)
- dataset D ($size(D)$ denotes the number of samples in D)
- dataset size to trigger training n_D
- number of training iterations k_{bc}

$D \leftarrow \emptyset$ ▷ initialise empty dataset
$t_{collect} \leftarrow 0$ ▷ remaining timesteps to collect advice set as 0
reuse allowed $\leftarrow$ *False* ▷ set advice reuse off by default
for training steps $t \in \{1, 2, \dots t_{max}\}$ **do**
  if $size(D) == n_D$ **then**
    Train G_ω for k_{bc} iterations ▷ behavioural cloning
  end if
  $a_t \leftarrow None$ ▷ set action as non-determined
  if *Env* is reset **then**
    $u \sim \mathcal{U}(0, 1)$ ▷ draw a number uniformly at random
    if $u < \varepsilon_{reuse}$ **then**
      reuse allowed $\leftarrow$ *True*
    else
      reuse allowed $\leftarrow$ *False*
    end if
  end if
  get observation $s_t \sim Env$ if *Env* is reset
  if $b > 0$ **then**
    $a_t \sim \pi_T$ ▷ collect advice
    add $\langle s_t, a_t \rangle$ to D
    $b \leftarrow b - 1$ ▷ decrement budget by 1
  end if
  if a_t is *None* **and** $\pi_S(a \mid s_t)$ is explorative **and** G_ω is trained **and** $G^\mu_\omega(s_t) < \tau_{reuse}$ **and** *reuse allowed* **then**
    $a_t \leftarrow \arg\max_a G_\omega(a \mid s_t)$ ▷ generate imitated advice
  end if
  if a_t is *None* **then**
    $a_t \sim \pi_S$ ▷ e.g., epsilon-greedy
  end if
  Execute a_t and obtain r_t, $s_{t+1} \sim Env$
  Update task-level model, e.g., DQN.
  $s_t \leftarrow s_{t+1}$
end for

5 EVALUATION DOMAIN

In order to have a significant complexity level as well as the challenges that are relevant to the deep RL methods in our experiments, we chose three Atari 2600 games from the commonly used Arcade Learning Environment (ALE) [2] as our evaluation domain:

- **Enduro:** The player controls a racing car in a long-distance track over multiple in-game days. In each day, if the player manages to pass a certain number of other cars (200 in the first day, 300 in the rest) in the race, it gets to advance to the next day. Progression during the days is visualised by different colour schemes that resemble the day-night cycle. Furthermore, there are other factors of seasonal events that affect the gameplay such as fogs and icy patches appearing on the road. Finally, as the days progress, the game increases in difficulty due to the other cars' behaviour becoming more aggressive.
- **Freeway:** In this game, the objective is to cross a chicken across a highway comprised of ten lanes with vehicles traversing in different directions and speeds. If the player hits the cars along the way, it gets pushed back towards starting point. Every time the player manages to reach the goal, it acquires a reward and gets teleported back at the starting point.
- **Pong:** This game consists of two paddles on the each side of the screen and a ball traversing around. The paddles are controlled by one player each. The players must hit the incoming balls to avoid them passing through their side as well as getting them thrown back at the opponent. If a player lets the ball pass through the gap behind its paddle, the opponent earns 1 point. In the single-agent variant of this game used in our study, the player controls the right side paddle while the other one is controlled by a built-in AI.



(a) Enduro (b) Freeway (c) Pong

Figure 1: Screenshots from the games of Enduro (a), Freeway (b) and Pong (c) within the Arcade Learning Environment.

Each of these games has an observation size of $160 \times 210 \times 3$, representing RGB images of the game screen that are produced at 60 frames per second (FPS). To make experimenting in these games computationally tractable, we employ some preprocessing steps that are also followed commonly in other studies [5]. First of all, each observation is made greyscale and resized down to the size of 80×80×1. Since the games run at a high FPS, the frame that is shown to the player is set to be only every 4th one (which is composed of the maximum pixel values of previous 3 frames), and the player's actions are repeated for the skipped frames. Moreover, since these games contain a fair amount of partial observability, such as the direction of the ball in Pong, the final form of the observation to be perceived by the player is made to be a stack of 4 pre-processed frames with a size of $84 \times 84 \times 4$ (which contains the information of the most recent 16 actual game frames). In order to deal with the varying range of reward scales and reward mechanisms within these games, every reward obtained in a single step in the game is clipped to be in $[-1, 1]$. Finally, every game episode is limited to last for maximum 108k frames, which corresponds to approximately 30 minutes of actual gameplay time in real-life.

Another set of modifications also take place to introduce more stochasticity within the games to turn them into more challenging

RL tasks. In the beginning of the games, the player takes no-op actions for a random number of times in $[0, 30]$, to simulate the effect of having different initial states. Additionally, with a probability of 0.25, the actions executed by the player are repeated for an additional step, which is referred to as *sticky actions*.

6 EXPERIMENTAL SETUP

The goal of our experiments[1] is to demonstrate that it is possible to generalise the teacher advice to the unseen yet similar states with our method, and that it is an effective way of improving performance of action advising, in complex domains especially. Therefore we choose the games described in Section 5 as our test-beds. The set of the student agent variants we compare are listed as follows:

- **No Advising (None)**: No action advising procedure is followed; the student learns as normal.
- **Early Advising (EA)**: The student follows early advising heuristic to distribute its advising budget. Specifically, the teacher is queried for an advice at every step until the budget runs out.
- **Early Advising with Advice Reuse via Imitation (AR)**: The student follows our proposed strategy (Section 4) combined with early advising heuristic. It starts off by greedily asking for advice until its budget runs out; then, it activates its behavioural cloning module to imitate and reuse the previously collected advice in the remaining exploration steps.

All student agent variants employ the identical task-level RL algorithm which is DQN with double Q-learning and dueling networks enhancements, and ϵ-greedy policy as the exploration strategy. The convolutional neural network structure within the DQN in input-to-output order is as follows: 32 8×8 filters with a stride of 4, 64 4×4 filters with a stride of 2, 64 3×3 filters with a stride of 1, followed by a fully-connected layer with 512 hidden units and multiple streams that add up in the end (dueling). Additionally, the student agent variant AR also incorporates a behaviour cloning module, which is a neural network with an identical structure minus the dueling stream. All the layer activations are set to be ReLU. The hyperparameters are tuned prior to experiments and kept the same across all experiments can be seen in Table 1.

In this teacher-student setup, we also need a teacher from which the student can get good quality action advice. For this purpose, we trained a DQN agent for each of these games for 10M steps (40M actual game frames) to achieve a competent level performance in each.

The experiments are conducted by executing every student variant through a learning session 3M steps (12M actual game frames) for every game. The learning steps are kept relatively small compared to the teacher training since it is expected for the students to achieve high performance much quicker with the aid of advice. Through the learning sessions, the agents are also evaluated at every 25k^{th} step in a separate instance of the environment for 10 episodes. During evaluation, any form of exploration and teaching is disabled in order to assess the actual proficiency of the students.

In terms of action advising setup, we set the action advising budget as 10k steps which corresponds to only approximately 0.3% of the interactions in a learning session and also to almost one third of a full game episode (27k steps). Besides the budget, our proposed method AR also uses some additional hyperparameters which were tuned prior to the full length experiments and are kept the same across every game. The dataset size n_D to train G_ω is set as 10k which is the action advising budget as we employ early advising prior to behavioural cloning training. The number iterations to train G_ω is set as 50k. Episodic advice reuse probability ε_{reuse} is set as 0.5 meaning that the student will follow G_ω in half the episodes (in the appropriate states). Finally, advice reuse uncertainty threshold τ_{reuse} is set as 0.01 (determined empirically) and kept the same across all games. In the experiments with AR, we also record the actual advice actions generated by the teacher at every step (not seen by the student) to have access to the ground-truth values to measure the accuracy of the behavioural cloning module. Every particular experiment case is repeated and aggregated over 3 different random seeds.

Table 1: Hyperparameters used in the student's DQN (top section) and Behaviour Cloning Network (bottom section).

Hyperparameter name	Value
Replay memory initial size and capacity	50k, 500k
Target network update period	7500
Minibatch size	32
Learning rate	625×10^{-7}
Train period	4
Discount factor γ	0.99
ϵ initial, ϵ final, ϵ decay steps	1.0, 0.01, 500k
Minibatch size	32
Learning rate	0.0001
Dropout rate	0.2
# of forward passes to assess uncertainty	100

7 RESULTS AND DISCUSSION

The results of our experiments are presented in Figures 2,3 and Table 2. Figure 2 contains the plots for the evaluation scores obtained by None, EA and AR modes of the student in the games of Enduro, Freeway, Pong. In Figure 3, the plots of the advice reuse trends of AR in this set of games are displayed as cumulatively (top row) and in every 100 steps windows (bottom row). These plots are limited to the first 500k steps to only consider the exploration stage determined by the agent's ϵ-greedy schedule. Purple lines here represent all advice reuses combined, while the green lines indicate only the correctly imitated (in terms of being equal to the ground-truth teacher advice) advice pieces. These results are also reported in Table 2 in the numerical form where the evaluation scores are broken down in two parts of final value and area-under-the-curve, which represent the final agent performance and the learning speed, respectively. Furthermore, the table also contains the total number of exploration steps taken, as well as the percentage of the number of reused advice in the exploration steps and the percentage of correctly imitated advice in total number of reused advice (denoted in parentheses).

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

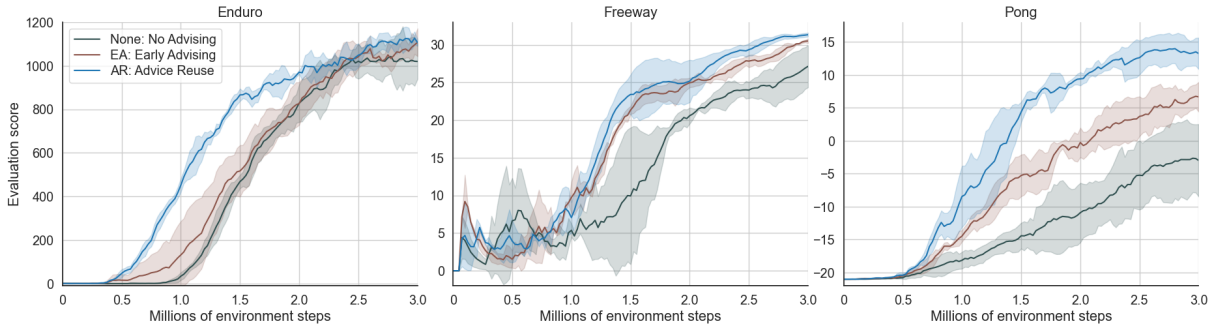


Figure 2: Evaluation scores of the student variants None, EA, AR obtained in the Atari games of Enduro (leftmost column), Freeway (middle column), Pong (rightmost column) aggregated over 3 runs. Shaded areas show the standard deviation across the runs.

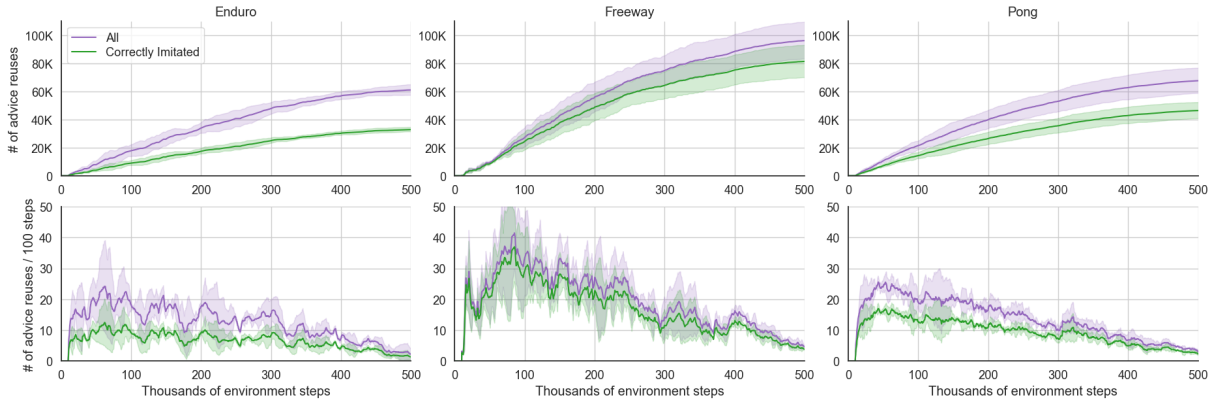


Figure 3: Number of advice reuses performed by the student with AR mode in the Atari games of Enduro (leftmost column), Freeway (middle column), Pong (rightmost column) over 3 runs, plotted cumulatively (top row) and in every 100 steps (bottom row). Purple lines represent the number of all advice reuses while the green lines represent the number of correctly imitated ones among these. Shaded areas show the standard deviation across the runs.

In the evaluation scores, we see different outcomes in each of these games. In Enduro, we see that AR provides a significant amount of jump start and performs the best in terms of learning speed while being far ahead of EA and None which are quite similar. When it comes to the final performance however, while EA and AR both outperform None, they do not differ much from each other. In Freeway, EA and AR perform very similarly in terms of learning speed and final performance with AR being slightly ahead of EA. However, they outperform None significantly. This shows that it matters to be advised initially, though their repetitions may not always yield much acceleration in learning. Finally, in Pong, we see a great difference between the performances in every aspect. Our AR comes out far ahead than its closest follower EA both in terms of final score and learning speed. This is an example of how getting a very little advice in the beginning as well as repeating them across further explorative actions can cause a great impact on the learning. Overall, AR manages to be the best in every game and suffers no performance loss even with high advice utilisation (as high as 104k in Freeway) which was shown to be harmful to learning in previous studies. Even though its performance boost over None seems to be not huge in every scenario, it should be noted that this is the case of it being combined with EA baseline. With more complicated methods, AR can be capable of training its imitation learning module with a more diverse set of experience and therefore, have a larger coverage which can potentially yield superior performance.

The task-level performance of our approach is affected primarily by two factors: the accuracy of advice imitation and its coverage/usage in the remainder of the exploration steps (the process of reusing). Therefore, we also analyse the advice reuse statistics of AR to form links between these outcomes. First of all, it should be noted that the decreasing trend in these plots is caused by the ϵ-greedy annealing. Enduro is the game with the smallest advice reuse rate as well as the lowest imitation accuracy. This is possibly because of the game episodes lasting long regardless of the agent's performance, which is likely to reduce the proportion of the familiar states according to the behavioural cloner. In Freeway, we observe a fairly high advice reuse rate with high accuracy of imitation. However, this is not reflected in the performance difference obtained versus EA, unlike in Enduro and Pong. Finally, in

Table 2: Final and area-under-the-curve (AUC) values of evaluation score plots (Figure 2), the number of exploration steps, the number of advice reuses (all and correctly imitated) of None, EA, AR student modes obtained in the Atari games of Enduro, Freeway, Pong aggregated over 3 runs. The numbers denoted by ± indicate standard deviation. The numbers in the parentheses show the percentage of reused advices in the exploration steps (in the column titled "All") and the percentage of correctly imitated advices in total number of reused advices (in the column titled "Correctly Imitated").

Game	Mode	Evaluation Score		# of Exp. Steps	# of Advice Reuses	
		Final	AUC ($\times 10^2$)		All	Correctly Imitated
Enduro	None	1021.54 ± 79.5	570.61 ± 38.4	326939 ± 92.1	—	—
	EA	1095.55 ± 45.9	616.29 ± 58.1	326753 ± 220.9	—	—
	AR	**1112.79 ± 16.6**	**782.98 ± 8.4**	326889 ± 230.5	67198 ± 3061.0 (20.55%)	36534 ± 1210.9 (54.44%)
Freeway	None	26.87 ± 2.3	15.73 ± 1.7	326872 ± 199.9	—	—
	EA	30.44 ± 0.2	20.31 ± 0.4	327158 ± 6.2	—	—
	AR	**31.28 ± 0.2**	**21.52 ± 1.0**	326778 ± 494.4	104770 ± 12522.2 (32.05%)	88829 ± 10950.5 (84.74%)
Pong	None	−2.78 ± 4.3	−16.24 ± 2.6	326744 ± 25.2	—	—
	EA	6.66 ± 1.6	−8.83 ± 0.4	326872 ± 199.9	—	—
	AR	**13.35 ± 1.7**	**−1.36 ± 1.0**	326933 ± 371.2	72581 ± 7615.7 (22.20%)	49538 ± 4853.8 (68.32%)

Pong, where the performance improvement is the most significant, advice reuse ratio seems to be similar to Enduro, but with far higher imitation accuracy.

Clearly, as we see from all these results combined, we can say that it is definitely a viable idea to extend the teacher advice over future states through imitation since this can be achieved with relatively high accuracy. However, even when we have access to these imitated competent policies, it is still non-trivial to construct a *good* exploration policy. While a higher advice reuse rate produces a more consistent exploration policy with less random dithering, it also has the risk of limiting the sample diversity in the replay memory, which can be problematic especially if the imitation quality is also poor. As long as the reuse amount does not get excessively high, it is safe to have the imitation learning accuracy around these reported levels, which makes tuning the uncertainty threshold straightforward. This is especially important for the realistic applications where it is not possible to access the tasks to tune such hyperparameters beforehand.

Finally, we also analyse our approach's computational burden, which may be the primary concern when adopting it. Specifically, it involves two extra operations: behavioural cloning network training and uncertainty estimations. The former happens only once in the beginning and therefore is negligible. The uncertainty estimations that require multiple forward passes (which is 100 in our experiments) happens in every exploration step and was found to cause a maximum of 2× slowdown in our experiments. Considering that the exploration steps only spans approximately 10% of a learning session, we can expect the runs to be taking at most 10% longer in total when AR is employed in a similar setting to ours; and, this becomes even smaller when the learning sessions last longer in terms of the total number of environment steps. Clearly, this is a small setback considering the sample efficiency benefits our method brings.

8 CONCLUSIONS AND FUTURE WORK

In this study, we developed an approach for the student to imitate and reuse advice previously collected from the teacher. This is the first time such an approach has been proposed in deep reinforcement learning (RL). In order to do so, we followed an idea similar to behavioural cloning, employing a separate neural network that is trained with the advised state-action pairs via supervised learning. Thus, this module can imitate the teacher's policy in a generalisable way that lets us apply it to the unseen states. We also incorporated a notion of epistemic uncertainty via dropout in this neural network to be able to limit the imitations to the states that are similar to the advice collected states.

The results of the experiments in 3 Atari games have shown that it is a feasible idea to accurately generalise a small set of teacher advice over unseen yet similar states in future. Furthermore, our approach of employing behavioural cloning was found to be a successful way of achieving this, as it yielded a considerably high accuracy of imitation in multiple games. Additionally, reusing these self-generated advice across the exploration steps provided significant improvements in the learning speeds and the final performances without any over-advising induced performance deterioration. Therefore, our method can be considered as a promising enhancement to the existing action advising methods, especially since it is also very straightforward to implement and tune, with only a small computational burden. Finally, it was also seen that utilisation of such imitated advice policies to construct good quality exploration is non-trivial and requires further investigation.

Our study lies at the intersection of action advising and exploration in RL and can be extended in various interesting ways. It is unclear how far the different qualities of imitation and reuse rates can affect performance in one particular game; it will be a worthwhile study to analyse these. Furthermore, evaluating the advice in terms of its contribution to learning progress is a promising direction to take.

REFERENCES

[1] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara J. Grosz. 2016. 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*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 804–811.

[2] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* 47 (2013), 253–279.

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. DotA 2 with Large Scale Deep Reinforcement Learning. *CoRR* abs/1912.06680 (2019).

[4] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. 2018. Exploration by Random Network Distillation. *CoRR* abs/1810.12894 (2018).

[5] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. 2018. Dopamine: A Research Framework for Deep Reinforcement Learning. *CoRR* abs/1812.06110 (2018).

[6] Si-An Chen, Voot Tangkaratt, Hsuan-Tien Lin, and Masashi Sugiyama. 2018. Active Deep Q-learning with Demonstration. *CoRR* abs/1812.02632 (2018).

[7] Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*. ACM, 1100–1108.

[8] Felipe Leno da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2020. 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, 5792–5799.

[9] Felipe Leno da Silva, Garrett Warnell, Anna Helena Reali Costa, and Peter Stone. 2020. Agents teaching agents: a survey on inter-agent transfer learning. *Auton. Agents Multi Agent Syst.* 34, 1 (2020), 9.

[10] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. 2015. The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, NIPS*.

[11] Anestis Fachantidis, Matthew E. Taylor, and Ioannis P. Vlahavas. 2019. Learning to Teach Reinforcement Learning Agents. *Machine Learning and Knowledge Extraction* 1, 1 (2019), 21–42.

[12] Yarin Gal and Zoubin Ghahramani. 2016. 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 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 1050–1059.

[13] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. 2018. Reinforcement Learning from Imperfect Demonstrations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*.

[14] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 30th innovative Applications of Artificial Intelligence 8th AAAI Symposium on Educational Advances in (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3215–3222.

[15] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2018. Deep Q-learning From Demonstrations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3223–3230.

[16] Ercüment Ilhan, Jeremy Gow, and Diego Pérez-Liébana. 2019. Teaching on a Budget in Multi-Agent Deep Reinforcement Learning. In *IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019*. 1–8.

[17] Ercüment Ilhan and Diego Pérez-Liébana. 2020. Student-Initiated Action Advising via Advice Novelty. arXiv:2010.00381

[18] Dong-Ki Kim, Miao Liu, Shayegan Omidshafiei, Sebastian Lopez-Cot, Matthew Riemer, Golnaz Habibi, Gerald Tesauro, Sami Mourad, Murray Campbell, and Jonathan P. How. 2020. Learning Hierarchical Teaching Policies for Cooperative Agents. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 620–628.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

[20] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 3307–3317.

[21] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. 2019. Learning to Teach in Cooperative Multiagent Reinforcement Learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 6128–6136.

[22] Dean Pomerleau. 1991. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Comput.* 3, 1 (1991), 88–97.

[23] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[24] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

[25] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[26] Matthew E. Taylor, Nicholas Carboni, Anestis Fachantidis, Ioannis P. Vlahavas, and Lisa Torrey. 2014. Reinforcement learning agents providing advice in complex video games. *Connect. Sci.* 26, 1 (2014), 45–63.

[27] Lisa Torrey and Matthew E. Taylor. 2013. 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*. 1053–1060.

[28] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 2094–2100.

[29] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. StarCraft II: A New Challenge for Reinforcement Learning. *CoRR* abs/1708.04782 (2017).

[30] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. 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 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 1995–2003.

[31] Yusen Zhan, Haitham Bou-Ammar, and Matthew E. Taylor. 2016. 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*. 2315–2321.

[32] Changxi Zhu, Yi Cai, Ho-fung Leung, and Shuyue Hu. 2020. Learning by Reusing Previous Advice in Teacher-Student Paradigm. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 1674–1682.

[33] Matthieu Zimmer, Paolo Viappiani, and Paul Weng. 2014. Teacher-Student Framework: A Reinforcement Learning Approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*.