An Enhanced Advising Model In Teacher-Student Framework Using State Categorization

Daksh Anand * [1], **Vaibhav Gupta** * [1], **Praveen Paruchuri** [1], **Balaraman Ravindran** [2]
[1] Machine Learning Lab, IIIT Hyderabad
[2] Robert Bosch Center for Data Science and AI, IIT Madras
daksh.anand@research.iiit.ac.in, gupta.vaibhav@research.iiit.ac.in, praveen.p@iiit.ac.in, ravi@cse.iitm.ac.in

Abstract

The teacher-student framework aims to improve the sample efficiency of RL algorithms by deploying an advising mechanism in which a teacher helps a student by guiding its exploration. Prior work in this field has considered an advising mechanism where the teacher advises the student about the optimal action to take in a given state. However, real-world teachers can leverage domain expertise to provide more informative signals. Using this insight, we propose to extend the current advising framework wherein the teacher would provide not only the optimal action but also a qualitative assessment of the state. We introduce a novel architecture, namely **A**dvice **R**eplay **M**emory (**ARM**), to effectively reuse the advice provided by the teacher. We demonstrate the robustness of our approach by showcasing our experiments on multiple Atari 2600 games using a fixed set of hyper-parameters. Additionally, we show that a student taking help even from a suboptimal teacher can achieve significant performance boosts and eventually outperform the teacher. Our approach outperforms the baselines even when provided with comparatively suboptimal teachers and an advising budget, which is smaller by orders of magnitude. The contributions of our paper are 4-fold (a) supplementing student's knowledge by providing the state category (b) introduction of ARM to effectively reuse the advice throughout learning (c) ability to achieve significant performance boost even with a coarse state categorization (d) enabling the student to outperform the teacher.

Introduction

Recent advances in deep learning and their combination with conventional Reinforcement Learning (RL) techniques have shown promising results in solving complex decision-making problems. Although there have been many successful RL applications, scaling up to large state spaces remains a challenge. In many real-world settings like autonomous driving (Michels, Saxena, and Ng 2005) and robot navigation (Matarić 1997), learning time is very expensive. Transfer Learning (Taylor and Stone 2009) alleviates this problem by using domain expertise to accelerate the learning speed.

The Teacher-Student framework (Torrey and Taylor 2013) is one such paradigm, where a domain expert (teacher) helps accelerate the student's learning by providing advice on the action to take in a given state. Recently, there have been several enhancements to this framework. Many extensions model the teacher as an RL agent solving an MDP to make the student learn as fast as possible (Fachantidis, Taylor, and Vlahavas 2019; Omidshafiei et al. 2019; Zimmer, Viappiani, and Weng 2014). In all these approaches, the teacher transfers the information in the form of action advice.

Although such guided exploration helps the student to converge faster, real-world teachers typically do not guide the student only about the best possible action in a particular situation. Instead, they may also provide a qualitative assessment of how preferable the given situation is (Gupta et al. 2019). Such richer advising can help the student relate better to the environment by learning about favorable states and making better use of this knowledge in the future. Using this insight, we believe that all the above advising frameworks can benefit via better utilization of the teacher's knowledge. In particular, the proposal is to make the information provided by the teacher richer by advising the student not just on the best action to take in a state but also about how promising that state is.

One way the teacher can provide such information is by partitioning the state space into a set of (qualitative) categories. On being queried, along with the optimal action to take, the teacher can inform the student about the state's category as per the teacher's estimate. To utilize the advice of this nature, the student will have to process the advice into a form that is compatible with its learning algorithm. The student is an RL agent, learning to optimize its performance using numerical feedback. Hence we assume that it can interpret and adapt to numerical values. Therefore, the student would need to map the space of qualitative classes to a numerical space using a well-defined mathematical function. Since many RL agents use value function approximators to model their policies, it would be beneficial to define this function as an approximation to V(s) (the state's value). In the case of Q-learning agents, this would directly provide the value of the optimal action, which is $Q(s, a)$ where a is the optimal action in the given state s. The student can use these mapped values to guide its learning.

In this paper, we use these values to move the student's Q-Network in the right direction. This means, the student moves its $V(s)$ ($Q(s, a)$ for the optimal action a) values in the direction of this approximate value. Due to the spatial

*Both authors contributed equally

nature of updates in Q-learning, this value will be gradually propagated to all the states near s. Hence if the student obtains the approximate values of multiple states in the early stages, it can have a better idea of the prospects of following different trajectories, allowing it to make more informed decisions. In order to effectively reuse the information provided in the advice, we introduce a novel architecture namely the **A**dvice **R**eplay **M**emory (**ARM**). ARM stores the advice tuples so that the agent can repeatedly relive these experiences over time and refresh what it has learned from the advice.

It may not always be feasible in real-world scenarios to have access to an optimal teacher for complex tasks. Even when used with sub-optimal teachers, our approach allows students to achieve significant performance boosts and eventually outperform them. Moreover, real-world teacher-students can have limited advice budgets due to communication constraints. Our approach considerably outperforms the baselines, even when restricted to an advice budget smaller by an order of magnitude.

Most of the existing work in the teacher-student framework follows an interactive setting, where the student can potentially query the teacher at any time during the learning period. However, in certain settings, the teacher might not be available to guide the student at all times. Hence, we present an interesting "non-interactive learning" setting where the teacher can compose a batch of advice tuples and provide it to a student at the beginning of its learning period. From thereon, even if the teacher goes offline or does not pay any attention, the student might still be able to gain reasonable performance boosts. The unique design of our approach presents a natural way to work in such settings, as opposed to the existing teacher-student paradigms.

We also investigate an interesting use case where the student, while learning on its own, is incapable of achieving the scores attained by the teacher. It can be because of the better network architecture or higher resources of the teacher. We believe that this use case is particularly useful in real-world situations. The student can be a hand-held device with limited resources, which can benefit from querying a server (the teacher) with higher resources.

We begin with an RL agent in the teacher's role, but we prefer advising mechanisms that can potentially be used with human teachers as well. It limits us to human-expressible teaching methods since they can generally provide only a coarse view of the environment. It also prevents the student from starting with the teacher's knowledge or assuming any access to the teacher's internal architecture. Furthermore, it requires teachers to be able to guide students that may learn and perceive their environment differently.

We demonstrate the performance of our approach on three domains from the Arcade Learning Environment (Bellemare et al. 2013), namely Qbert, Boxing and Seaquest. We believe that our approach can extend the existing Teacher-Student frameworks (Amir et al. 2016; Fachantidis, Taylor, and Vlahavas 2019; Torrey and Taylor 2013; Zimmer, Viappiani, and Weng 2014), where only the action is communicated as advice.

Related Work

In episode sharing, the teacher communicates successful episodes to accelerate a student's learning (Tan 1993). A confidence-based approach was proposed in (Chernova and Veloso 2007), where the student learns only from expert demonstrations without receiving any reward signal from the environment. Instead of giving teacher's synthesized information, these approaches simply provide crude information like teacher's trajectories. Reward shaping uses an artificial reward signal to guide the agent in a more controlled fashion (Ng, Harada, and Russell 1999). However, it requires detailed knowledge of the states' potential, which might be hard to represent in the form of a shaped reward (Randløv and Alstrøm 1998). In contrast, our approach only requires a coarse categorization of a limited number of states.

In the teacher-student framework, the teacher helps to accelerate the student's learning by providing advice on the action to take in a given state. In the student-initiated advising paradigm (Clouse and Utgoff 1992), the student asks the teacher for advice whenever its confidence in a state is low. The teacher responds with the optimal action to take in that state. Torrey considered a teacher-initiated advising paradigm (Torrey and Taylor 2013) with a limited advice budget. Here, the teacher continuously monitors the student's decisions until the advice budget runs out. Amir proposed a jointly-initiated advising mechanism (Amir et al. 2016), where the student decides when to ask for teacher's advice based on one of the student-initiated approaches. The teacher then decides whether to provide advice based on one of the teacher-initiated advising approaches.

This framework was further extended by relaxing the need to have expert advisors in the MAS setting, where simultaneously learning agents advise each other, and the roles of teacher and student are interchangeable (Da Silva, Glatt, and Costa 2017). An advice reuse strategy was proposed wherein the student stores all the states where the teacher advised the action to take (Zhu et al. 2020). This approach fails to generalize the received knowledge to unseen states since the previously advised action is reused only if the same state is revisited. A Q-Values sharing framework was proposed wherein the advisee simply copies the adviser's Q-Values into its Q-Table when advised in a given state (Zhu et al. 2019). The bookkeeping nature of these approaches limits them to work with simple domains that can be discretized via techniques like tile coding. There have also been efforts to model teachers as RL agents that try to make the student learn as fast as possible (Fachantidis, Taylor, and Vlahavas 2019; Zimmer, Viappiani, and Weng 2014). The approach of modeling teachers as RL agents has also been extended to cooperative multi-agent settings (Omidshafiei et al. 2019).

In contrast to our approach, the advising mechanisms mentioned above allow the teacher to advise the student solely about the optimal action to perform. Although Q-Values sharing explores a form of richer advising, but the strict requirement of having Q-Table makes it infeasible for complex domains. All these approaches can potentially benefit by utilizing the teacher's knowledge better in the form of richer advising, i.e., the teacher provides a qualitative assessment of the state along with the optimal action to take.

Background

The following subsections provide brief background on standard RL algorithms and the teacher-student framework.

Q-Learning and Deep Q Network (DQN)

One of the most popular algorithms for solving sequential decision problems is Q-learning (Watkins and Dayan 1992; Watkins 1989). It tries to estimate the optimal value of every state-action pair $Q(s, a)$. The optimal value of an action a in a state s, $Q^*(s, a)$, is defined as the expected cumulative sum of future rewards on taking a in s and following the optimal policy thereafter.

DQN (Mnih et al. 2015) combines the representation learning power of deep neural networks with the Q-Learning objective to approximate the Q-value function. The off-policy nature of the algorithm ensures that DQN can avoid correlated updates, using an experience replay memory (Lin 1992). The loss function for DQN is given by:

$$L(\theta) =_{s,a,r,s'} [(y_{DQN} - Q(s, a; \theta))^2] \quad (1)$$

with:

$$y_{DQN} = (r + \lambda \max_{a'} Q(s', a'; \theta')) \quad (2)$$

Here, L represents the expected TD error corresponding to current parameter estimate θ. θ' and θ represent the parameters of the target network and the online network respectively. The gradient descent step is as follows:

$$\nabla_\theta L(\theta) =_{s,a,r,s'} [(y_{DQN} - Q(s, a; \theta))\nabla_\theta Q(s, a; \theta)] \quad (3)$$

Teacher Student Framework

The teacher-student framework (Clouse and Utgoff 1992; Torrey and Taylor 2013) aims at accelerating a student's training through advice on action to take from an experienced teacher. The student queries the teacher in a given state, and the teacher advises it about the action to take in that state. The student simply performs the action suggested by the teacher.

In the student initiated advising (Clouse and Utgoff 1992), the student decides when to ask the teacher for advice. There are many heuristics like *Ask Uncertain* where the student asks for advice whenever its confidence in a state is low. If in a given state, the expected rewards associated with all the actions are close to each other, then the student is indifferent about which action to choose. The confidence metric for the student in a given state is given in equation (4).

In the teacher-initiated advising (Torrey and Taylor 2013), the teacher identifies when to advise the student. For identifying when to advise, the teacher used several heuristics like *Mistake Correcting Advising* and *Importance Advising* i.e., advising when state importance is more than a threshold. Intuitively, a state is considered important if taking a wrong action in that state can lead to a significant decrease in future rewards. In the case of agents using TD algorithms, the state importance metric is given in equation (5).

$$C(s) = \max_a Q_{student}(s, a) - \min_a Q_{student}(s, a) \quad (4)$$

$$I(s) = \max_a Q_{teacher}(s, a) - \min_a Q_{teacher}(s, a) \quad (5)$$

In jointly Initiated Strategies (Amir et al. 2016), the student decides when to ask the teacher for advice based on one of the student-initiated heuristics. Then, the teacher decides whether to provide advice based on one of the teacher-initiated advising heuristics.

Approach

We propose to extend the existing Teacher-Student approaches by making the teacher advise the student not just on the best action to take in a state but also provide a qualitative assessment of how promising that state is. As discussed earlier in the Introduction, since the student is an RL agent using DQN, we need to project these qualitative assessments to a numerical form representing that state's value.

One way in which the teacher can provide qualitative information is by partitioning the state space into a set of qualitative categories. State characterization can be done based on the quality of the outcomes achieved by following the best possible sequence of actions starting from that state. It naturally induces a relative ordering between the states based on their categories. Since the teacher is an RL agent, it can estimate the maximal (V_{high}) and the minimal (V_{low}) long-term returns that its policy can achieve starting from any random state.

Given these global bounds (V_{high}, V_{low}) and the category of a particular state s, the student can map it to a numerical value. Let there be a set of n categories $C = \{c_1, c_2....c_n\}$, where $c_i \succ c_{i+1}, \forall_{i=1...n-1}$ i.e. i^{th} category is preferable over (or has higher associated long term reward than) all the succeeding categories. We define a function $\psi(c_i)$ to project from the space of these qualitative categories to a numerical space. The projected values should be an approximate representation of $V(s)$, the value of the state as given below:

$$\psi(c_i) = \frac{[\omega * i] + [\omega * (i - 1)]}{2} \quad (6)$$

Here, $\omega = (V_{high} - V_{low})/n$ i.e. the gap between 2 successive categories. ψ maps c_i to n equally spaced values from the range defined by V_{high} and V_{low}. The more preferred category is given a higher value. This mapped value equivalently gives us an approximation of the value of the most preferred action suggested by the teacher. For the rest of the paper, we assume that $\psi(s)$ denotes $\psi(c^s)$ where c^s is the category assigned to state s by the teacher.

Since $\psi(s)$ gives the approximate value for $Q(s, a)$ (where a the is most preferred action suggested by the teacher), it provides a reasonable estimate of the direction in which the agent should move its Q-network. Due to the high level of generalization over the state space in DQN, doing large updates in the direction of $\psi(s)$ might lead to overfitting and instability. Also, it would be wasteful to use these values only once and then forget about them. The student must also give proper importance to the experiences it gathers by following its own policy. It needs to smoothen the training over both the experiences due to its own policy and

the experiences due to the teacher's advice. To balance the learning of these experiences with the normal updates, the student needs to steadily move its network in the direction of these approximate values($\psi(s)$) throughout its learning.

To handle this, we introduce an additional replay memory called the **A**dvice **R**eplay **M**emory (**ARM**). It stores the advice tuples $(state, action, category)$ for every teacher's advice. The student can repeatedly relive the experiences due to the teacher's advice over time. Thus, the student's network needs to minimize an additional loss L' due to the ARM tuples. It is given by:

$$L'(\theta) =_{s,a} [(\psi(s) - Q(s,a;\theta))^2] \quad (7)$$

As typical to DQN agents, the student also has to minimize a loss L due to the tuples in D, the experience replay memory. This loss is the same as given in equation (1). To smoothen the training over both kinds of experiences, at every training step, the student randomly samples a mini-batch from each of these memories and computes the losses for them. The gradients for the losses L and L' are as follows:

$$\nabla_\theta L(\theta) =_{s,a,r,s'} [(y_{DQN} - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta)] \quad (8)$$

$$\nabla_\theta L'(\theta) =_{s,a} [(\psi(s) - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta)] \quad (9)$$

Due to the random initialization of DQN, initially, the state-action values predicted by the student would be somewhat random and would differ a lot from $\psi(s)$. It makes the loss L' computed due to the tuples from ARM quite large. On the other hand, due to the spatial nature of updates in DQN, the difference in the values predicted by the network and the expected targets $(r + max_a Q(s',a))$ would be smaller, making the loss L computed due to the tuples from D comparatively smaller. If we compute the combined loss for L' and L, L' would tend to dominate over L, and the network would almost neglect the updates due to transition tuples (s,a,r,s'). It can make the network unstable and lead to chaotic behavior. To ensure smooth learning of the network, we take a weighted average of L' and L. The combined loss and its gradient are given by:

$$L_{total} = (1-\alpha) * L + \alpha * L' \quad (10)$$

$$\nabla_\theta L_{total}(\theta) = (1-\alpha) * \nabla_\theta L(\theta) + \alpha * \nabla_\theta L'(\theta) \quad (11)$$

This weighted average makes our algorithm robust to the large differences that might be there in the predicted values and the values given by the teacher. The value of α, the *advice ratio*, was determined using a grid search. The sensitivity of our algorithm to the α parameter is presented in detail in the appendix.

Initially, the student leverages the teacher's knowledge to boost its performance, but eventually, its own policy would reach high quality and even outperform the teacher's. Therefore, the student needs to reduce the relative importance of the teacher's advice over time. To balance the effect of transferred knowledge and its own learning experiences, it starts decaying α. The student can use any decay policy δ, which can be as simple as exponential decay. As shown in our experiments below, the student following our approach can perform reasonably well even with sub-optimal teachers and eventually outperforms the teacher.

Algorithm 1 Advising Model using State Categorization

Require: Advice Replay Memory: ARM, Experience Replay Memory: D, advice ratio: α, α-decay policy: δ

Initialize: $s_1 \leftarrow getInitialState()$
for $t = 1, T$ **do**
  $a \leftarrow getAction(s_t)$
  Sample minibatch b_1 from ARM
  Sample minibatch b_2 from D
  $L' \leftarrow$ loss due to tuples in b_1 as per equation (7)
  $L \leftarrow$ loss due to tuples in b_2 as per equation (1)
  $L_{total} \leftarrow (1-\alpha) * L + \alpha * L'$
  Perform a gradient descent step on L_{total} according to equation (11)
  Execute a, observe reward r and next state s_{t+1}
  store transition (s_t, a, r, s_{t+1}) in D
  decay α using decay policy δ
end for

The complete algorithm is given in algorithm 1. The routine $getAction(s)$ returns the action to be taken by the student in the state s. If the teacher advises an action, the student appends the appropriate tuple to the Advice Replay Memory. $H_s(s)$ is the student's heuristic function to decide when to ask for advice. It returns $true$ if the student has to ask for advice in the state s. Similarly, $H_t(s)$ is the teacher's heuristic function to decide when to give advice. Figure 1 illustrates the overall setup.

Experimental Analysis

We experimentally showcase the effectiveness of our approach by extending Amir's framework (Amir et al. 2016). However, as discussed in the introduction, we believe that our approach can naturally extend other advising mechanisms as well as (Fachantidis, Taylor, and Vlahavas 2019; Stolle and Precup 2002; Torrey and Taylor 2013; Zimmer, Viappiani, and Weng 2014), where the action is communicated as advice. In particular, we show that richer advising about a state's category can significantly improve student's performance. Additionally, we demonstrate the effect of receiving advice from a sub-optimal teacher and show that the student can still achieve significant performance improvement and eventually outperform the teacher.

Hyper-parameters and Setup

For experimental purposes, for each game, we train an agent for 30 million steps and checkpoint it every 100,000 steps. After training, the checkpoint where the agent could attain 70% of the highest score is selected as the teacher for that particular game. The values of *advice ratio* α and the *batch size* were fixed to 0.01 and 8 respectively for this experiment. As discussed in the appendix, both of these define the

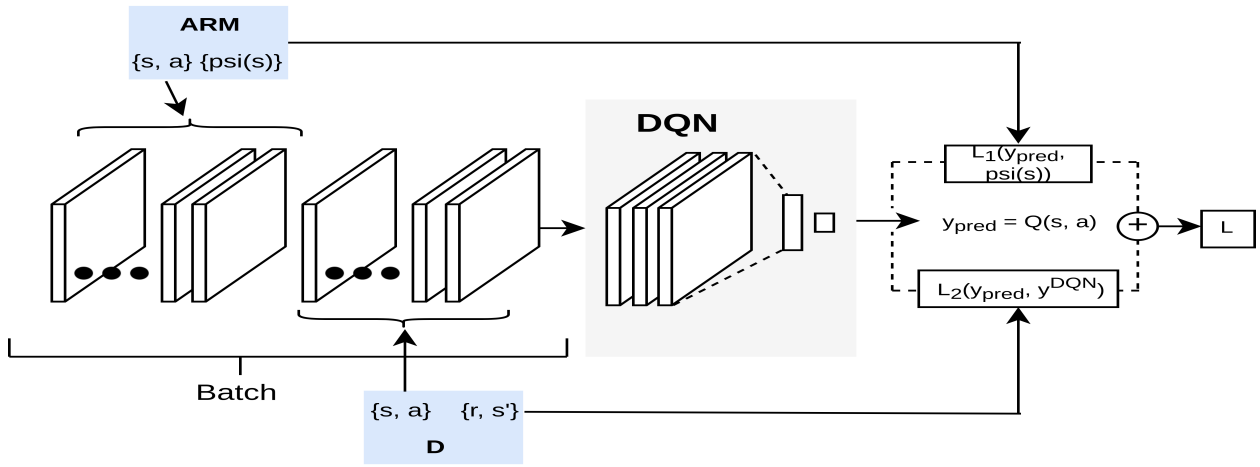


Figure 1: DQN augmented with Advice Replay Memory.

importance that the student gives to its own learning experience compared to the teacher's advice. As pointed above, our algorithm is flexible to work with any α-decay policy. In this paper, the student exponentially decays α by a factor γ every 100,000 training steps. For all the games, we fix γ to 0.99. In the appendix, a rigorous analysis of the student's performance is done across different values of γ. Our algorithm is stable with different values of γ, and in general, a relatively smaller value of γ is preferred with low-quality teachers. In figure 2, we analyze the effect of having different levels of suboptimal teachers on student's performance.

Algorithm 2 getAction(s)

Require: Student's heuristic function $H_s(s)$, Teacher's heuristic function $H_t(s)$, Student's policy $\pi_s(s)$, Teacher's policy $\pi_t(s)$, Teacher's heuristic for state categorization $\psi(s)$, Advice Replay Memory ARM

Output: $action$: action to take in the given state s

if $H_s(s) = true$ **then**
  if $H_t(s) = true$ **then**
    $action \leftarrow \pi_t(s)$
    $category \leftarrow \psi(s)$
    store advice $(s, action, category)$ in ARM
    return $action$
  end if
end if
$action \leftarrow \pi_s(s)$
return $action$

We use the *Ask Uncertain - Advice Important* heuristic for advice exchange, as given in Amir's work. It needs modeling hyper-parameters that vary across each game, which were determined experimentally as summarized in Table 1 of the appendix. These parameters dictate the approximate amount of advice exchanges. An advising budget of 10k was fixed for our approach, whereas Amir's approach was allowed to have a budget of 100k. The hyper-parameters specific to our approach were kept the same for all the games. The DQN architecture was kept the same as (Mnih et al. 2015). We use the Double-DQN algorithm (Van Hasselt, Guez, and Silver 2016) for training both the student and the teacher.

RL teachers might typically categorize a state based on the prospects associated with it. To simulate this behavior, we pre-trained an RL agent (using DQN) in the same environment. Two states are considered similar if they have similar prospects associated with them. Hence the teacher assigns the same category to the states having $V(s)$ in the same range of values, such that states with higher value belong to the more preferred category.

All the agents were trained for 30 million steps with the size of each training epoch being 40k steps. Each training epoch is followed by a testing epoch where the performance of the algorithm is averaged over 20 episodes. To smoothen the curves, all plots were averaged over five trials, and a running average of 15 was used in plotting the results. The solid curves correspond to the mean and the shaded region to the minimum and maximum returns over the five trials.

Effect of the number of State Categories: The *number of state categories* defines the granularity of information being provided to the student and was fixed to six in all the experiments. A significantly large value of n will provide a highly refined view of the environment to the student. In contrast, a small value might hamper its performance due to high approximation errors. As presented in the appendix, even a small value of n (four or six) is sufficient to provide a significant performance boost to the student.

Performance analysis w.r.t. different sub-optimal teachers: We analyze the student's performance following our approach for varying levels of sub-optimality of the teacher on Qbert. Over time, the advice provided by a sub-optimal teacher can start hampering the student's learning. The states might be miscategorized, or the teacher might have an inconsistent and suboptimal policy. The student may, therefore, need to assign higher importance to its own learning over time. We selected four different teachers with the following average scores: 3.5k, 5k, 6.5k, and 9k. Note that these scores for the teacher were obtained as an average

of 200 independent episodes. We also compare the performance of these students with the student using Amir's approach, which receives advice from an optimal teacher (i.e., an average reward of 9k).

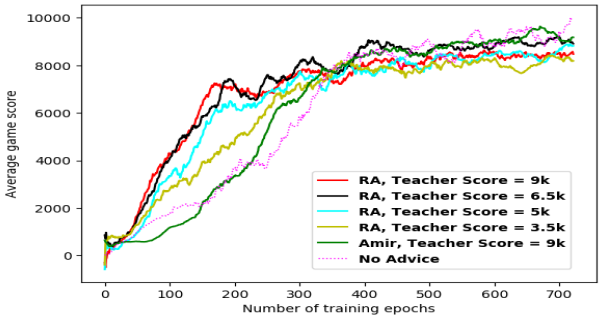


Figure 2: Performance analysis w.r.t. different sub-optimal teachers on Qbert. Running Avg of 40 was taken for clarity.

Figure 2 shows that a higher quality teacher's advice leads to a more significant boost in student's performance. An interesting outcome of this experiment is that even with a suboptimal teacher (with an average reward of 3.5k), our approach performs significantly better than Amir's approach even when the latter receives advice from an optimal teacher. Another key observation is that a sub-optimal teacher with 70% of the optimal score results in a similar performance boost to the student as an optimal teacher.

Figures 3 - 5 show a student's performance following our approach compared to a student to whom only actions were given as advice. We also show the performance of an agent who is learning without the teacher's help. The x-axis represents training epochs, while the y-axis represents the average episode reward based on the fixed policy at that point.

Advice Transfer Across Homogeneous Agents

In this subsection, both the teacher and the student use the same network architecture (Mnih et al. 2015) for their learning. Figures 3(a), 4(a), and 5(a) show that due to guided exploration, the student to whom only actions were advised has a boost in performance (in terms of learning rate) as compared to the agent which learns on its own. These figures showcase that the student to whom the teacher also provides a qualitative assessment of the state can significantly outperform both the above agents.

As clear from the graphs, our approach results in significant performance improvement in the initial phases. It is because the student (a typical Q-Learning agent using DQN) randomly samples the transitions tuples from D and updates its value function based on the loss function given in equation (1). The target value y_{DQN} depends on θ', making the target value of the network fluctuating in nature. Due to the random initialization of the neural network, the fluctuations are prominent at the start. However, since the category assigned to any state by the teacher does not change over time, $\psi(s)$ (the approximate value of the teacher's suggested action) remains fixed. Therefore, the student using our approach moves in the right direction from the start, resulting in greater performance improvement in the initial phases. Over time, all the students start to converge to an optimal policy, and the importance of these values becomes less significant. Hence, the difference in performance decreases.

Discussion about advising budget: As mentioned above, our approach's advising budget is fixed to an order of magnitude less than the baseline (10k and 100k respectively). Even with such a small budget, our approach can perform significantly better than the baseline. As shown in the appendix, similar performances can be observed for our approach, whether the student receives 400k advice or 7.5k advice. In contrast, the baseline demonstrates significantly poor performance when operated on such small budgets. It implies that our approach makes better use of every single advice tuple that it receives. It is because a student learning with our approach uses ARM to reuse all the advice information effectively. The student can repeatedly relive these experiences over time and refresh what it has learned from the advice. In contrast, the baseline uses advice only once and then wastefully forgets about it.

Advice Transfer Across Heterogeneous Agents

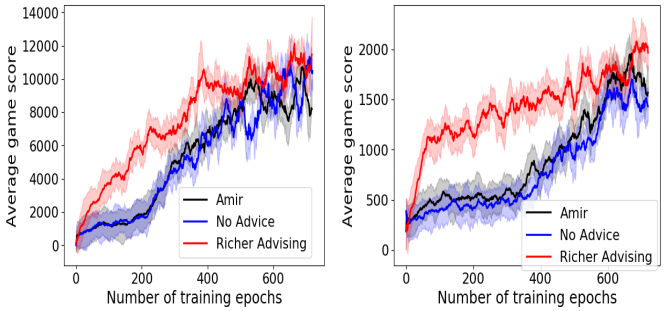
As discussed in Introduction, the use case where the teacher has a much more complex network than the student is particularly useful. Therefore, we use the models proposed by (Mnih et al. 2015) and (Mnih et al. 2013), as the underlying architectures for the teacher and the student, respectively. Figures 3(b) - 5(b) show similar trends as the graphs in Figures 3(a) - 5(a).

The key insight here is that advising leads to higher performance improvement than the case where the teacher and the student have similar networks. The model proposed in (Mnih et al. 2015) outperforms the model proposed in (Mnih et al. 2013), in terms of the convergence scores. Therefore, advising from a comparatively superior teacher becomes a lot more valuable for the student and results in much higher performance improvement.

Non-interactive Learning Setting

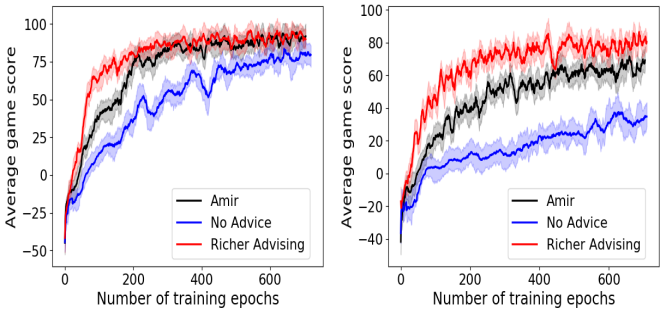
This experiment presents an interesting "non-interactive learning" setting where the teacher can compose a batch of advice tuples and provide it to the student at the beginning of its learning period. From thereon, even if the teacher goes offline or does not pay any attention, the student might still be able to gain reasonable performance boosts. The student treats this batch in the same way it uses advice tuples from ARM. One way of composing the batch is by randomly sampling the advice tuples from the Advice Replay Memory of any previously trained student. The experiments were performed on Boxing, on a spectrum of three different batch sizes having 7.5k, 15k, and 30k advice tuples.

It should be noted that due to the presence of ARM, our algorithm works naturally in such settings. However, other approaches do not have any efficient way of reusing the previously received advice i.e., advice received in any given state can only help in selecting the action in that particular state. Therefore, we do not have any suitable mechanism



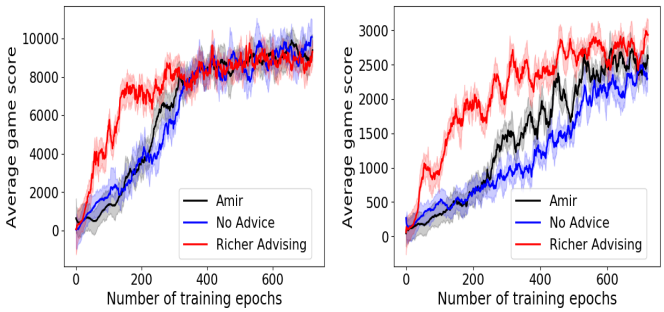
(a) homogeneous networks (b) heterogeneous networks

Figure 3: Seaquest training comparison.



(a) homogeneous networks (b) heterogeneous networks

Figure 4: Boxing training comparison.



(a) homogeneous networks (b) heterogeneous networks

Figure 5: Qbert training comparison.

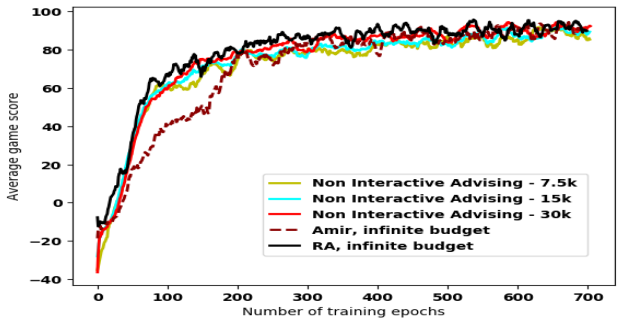


Figure 6: Analysis of performance in a non-interactive learning setting, shown on Boxing.

that enables them to work in this scenario. Figure 6 shows the performance of a student using our algorithm in this setting. The learning curve of the student using our approach who can query the teacher interactively is also plotted. We can observe that a student that receives a batch of 30k advice tuples performs marginally below the student who can query the teacher interactively. Furthermore, there is a gradual decrease in performance with a decrease in the size of the advice batch received. The key point to note here is that even a student with a small batch size of 7.5k tuples outperforms the baseline having an unlimited budget.

In our approach (in the interactive setting), the teacher helps the student in two forms, namely, (a) action suggestion and (b) state assessment. We analyze the individual impact of both forms of advice on the student's learning in Figure 4 of the appendix. Amir's approach solely relies on (a), and the "non-interactive learning" setting uses only (b). As can be seen in the figure, the student using Richer Advising (in the interactive setting) outperforms both the student using Amir's approach and the student using Non-Interactive advising. It implies that the combined effect of (a) and (b) is greater than either. Furthermore, the student learning in a Non-Interactive setting outperforms the student following Amir's approach. Hence the validation that the state assessment causes a more significant performance boost in a student than action suggestion.

Conclusion and Future Work

This paper investigates the role of richer knowledge transfer in the case of a Teacher-Student framework. Our experiments show that if the teacher provides additional advice in the form of a qualitative assessment of the state, our approach can outperform the advising mechanism being extended. Experiments show that our approach outperforms the baselines even when provided with comparatively suboptimal teachers and an advising budget, which is smaller by orders of magnitude. We introduce a novel architecture, namely ARM, to effectively reuse the teacher's advice and propose a way of giving proper attention to both (advising and normal learning) kinds of experiences.

To our knowledge, such richer advising has not been studied, especially in the context of the Teacher-Student framework. It would be interesting to examine the effectiveness of our approach in the case of simultaneous learning agents (Da Silva, Glatt, and Costa 2017) where the teacher-student roles are interchangeable, and there can be multiple teachers. In complex and hierarchical domains, the student might temporarily treat these approximate values as sub-goals. Another use case can be to use these values for reward shaping (Ng, Harada, and Russell 1999). The interesting use case where the teacher can give qualitative assessment only in a limited number of states might also be explored. In such scenarios, the student can find important states as in (Stolle and Precup 2002) and then query the teacher in these states.

References

Amir, O.; Kamar, E.; Kolobov, A.; and Grosz, B. J. 2016. Interactive Teaching Strategies for Agent Training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 804–811. AAAI Press. ISBN 9781577357704.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47: 253–279.

Chernova, S.; and Veloso, M. 2007. Confidence-based Policy Learning from Demonstration Using Gaussian Mixture Models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 233. ACM.

Clouse, J. A.; and Utgoff, P. E. 1992. A teaching method for reinforcement learning. In *Machine Learning Proceedings 1992*, 92–101. Elsevier.

Da Silva, F. L.; Glatt, R.; and Costa, A. H. R. 2017. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th conference on Autonomous Agents and Multiagent Systems*, 1100–1108.

Fachantidis, A.; Taylor, M. E.; and Vlahavas, I. 2019. Learning to teach reinforcement learning agents. *Machine Learning and Knowledge Extraction* 1(1): 21–42.

Gupta, V.; Anand, D.; Paruchuri, P.; and Ravindran, B. 2019. Advice Replay Approach for Richer Knowledge Transfer in Teacher Student Framework. In Elkind, E.; Veloso, M.; Agmon, N.; and Taylor, M. E., eds., *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, 1997–1999. International Foundation for Autonomous Agents and Multiagent Systems. URL http://dl.acm.org/citation.cfm?id=3331989.

Lin, L.-J. 1992. *Reinforcement Learning for Robots Using Neural Networks*. Ph.D. thesis, USA. UMI Order No. GAX93-22750.

Matarić, M. J. 1997. Reinforcement learning in the multi-robot domain. In *Robot Colonies*, 73–83. Springer.

Michels, J.; Saxena, A.; and Ng, A. Y. 2005. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 593–600. ACM.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* .

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Omidshafiei, S.; Kim, D.-K.; Liu, M.; Tesauro, G.; Riemer, M.; Amato, C.; Campbell, M.; and How, J. P. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6128–6136.

Randløv, J.; and Alstrøm, P. 1998. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, volume 98, 463–471.

Stolle, M.; and Precup, D. 2002. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, 212–223. Springer.

Tan, M. 1993. Multi Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the tenth International Conference on Machine Learning*, 330–337.

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(Jul): 1633–1685.

Torrey, L.; and Taylor, M. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi Agent Systems*, 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, volume 2, 5. Phoenix, AZ.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4): 279–292.

Watkins, C. J. C. H. 1989. *Learning from delayed rewards*. Ph.D. thesis, King's College, Cambridge.

Zhu, C.; Cai, Y.; Leung, H.-f.; and Hu, S. 2020. Learning by Reusing Previous Advice in Teacher-Student Paradigm. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1674–1682.

Zhu, C.; Leung, H.-f.; Hu, S.; and Cai, Y. 2019. A Q-values sharing framework for multiple independent Q-learners. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2324–2326.

Zimmer, M.; Viappiani, P.; and Weng, P. 2014. Teacher-student framework: a reinforcement learning approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*.