Let's Do the Time Warp Again: Human Action Assistance for Reinforcement Learning Agents

Carter B. Burn[1], Frederick L. Crabbe[1] and Rebecca Hwa[2]
[1]*Dept. of Computer Science, United States Naval Academy, Annapolis, MD, U.S.A.*
[2]*Dept. of Computer Science, University of Pittsburgh, Pittsburgh, PA, U.S.A.*

Keywords: Human-assisted Reinforcement Learning, Action-advice, Time-warp.

Abstract: Reinforcement learning (RL) agents may take a long time to learn a policy for a complex task. One way to help the agent to convergence on a policy faster is by offering it some form of assistance from a teacher who already has some expertise on the same task. The teacher can be either a human or another computer agent, and they can provide assistance by controlling the reward, action selection, or state definition that the agent views. However, some forms of assistance might come more naturally from a human teacher than a computer teacher and vice versa. For instance, a challenge for human teachers in providing action selection is that because computers and human operate at different speed increments, it is difficult to translate what constitutes an action selection for a particular state in a human's perception to that of the computer agent. In this paper, we introduce a system called Time Warp that allows a human teacher to provide action selection assistance to the agent during critical moments of the training for the RL agent. We find that Time Warp is able to help the agent develop a better policy in less time than an RL agent with no assistance and rivals the performance of computer teaching agents. Time Warp also is able to reach the results with only ten minutes of human training time.

1 INTRODUCTION

Reinforcement learning (RL) enables an agent to improve its ability to solve decision-making tasks as it interacts with a complex environment (Sutton and Barto, 1998). As it learns, the agent develops a policy to determine which actions to take in order to produce desirable outcomes in the environment. While most RL methods learn this policy through a trial-and-error process with no external guidance, some studies suggest that a *teacher* may provide worthwhile assistance to the agent still in training; that is, the teacher can help the learning agent to develop a better policy at a faster rate (Torrey and Taylor, 2013). The role of the teacher may be played by either a human or a computer agent that has already been trained in the domain. Although one might expect humans to be better teachers (since they have a broader view of the problem and can typically perform the tasks better than trained computer agents) results from previous work seem to suggest otherwise (Thomaz and Breazeal, 2006).

This paper re-examines the problem of assisting learning agents. We divide the space of types of assistance into three categories:

1. **Reward Feedback:** the teacher provides a reward or has control of a reward signal,
2. **Scenario Presentation:** the teacher configures sub-problems of the domain that the agent can train on to eventually learn a policy to the larger problem, and
3. **Action Advice:** the teacher provides advice to the agent by giving the correct action in important states, which enables the agent to find the high reward areas of the domain faster than traditional exploration.

In prior work, human teachers gave reward feedback while computer agent teachers provided action advice and scenario presentations, however, the entire space of types with both human and computer agent teachers has not been fully explored. We believe that human teachers are more suited to providing action advice than other forms of assistance, especially if they have no specialized expertise in either the problem domain or in machine learning (important for both scenario presentation and reward feedback). We think this because non-RL experts may accidentally

Burn, C., Crabbe, F. and Hwa, R.
Let's Do the Time Warp Again: Human Action Assistance for Reinforcement Learning Agents.
DOI: 10.5220/0010258700920100
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 1*, pages 92-100
ISBN: 978-989-758-484-8
Copyright © 2021 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

shape the agent's model of the reward function in unintended ways. This may explain some of the difficulties encountered by Thomaz and Breazal discussed in section 2.1. We hypothesize that with a mechanism for enabling humans to provide action advice, the advantages of computer teachers over human teachers would disappear. Our goal is to show that human teachers can provide action advice and help a learning agent find a better policy at a faster rate while spending less total time advising when compared to computer teachers and eliminate the redundancy of computer teachers who have already developed a policy.

A major challenge for human teachers in offering action advice is that humans operate at a much slower speed than their computer students, therefore it is difficult to pin-point the crucial states that the student needs action advice on. We address this by introducing the **Time Warp** guidance interface, which features a mechanism that allows the teacher to monitor the student as it learns, then freeze training and rewind time to the state that the teacher deems important in order to provide action advice from the desired state.

We tested Time Warp in an environment based on the classic arcade game Frogger. Experimental results suggest that human teachers are as effective in producing agents as the computer teachers. We see this effectiveness measured by the time to produce the policy and the performance of the agent. We also discover that the more exploration the human teacher provides while training eventually produces better-performing agents, rather than teachers that simply exploited the domain.

2 ASSISTED REINFORCEMENT LEARNING

A typical RL agent attempts to learn a solution to a task in a given environment through a trial-and-error process. While standard RL methods can produce well-performing agents, the training process may take a long time to converge. This is especially observable as the state space and the opportunities for explorations becomes large. Consequently, these methods may converge on policies that do not perform at an optimum level within a given state space. Assisted reinforcement learning helps to reduce wasteful exploration so that the trial-and-error process takes less time and produces better agents.

2.1 Forms of Assistance

There are several different ways in which a teacher might offer assistance to the training of an RL agent. We categorize prior work on assisted reinforcement learning into three main forms of assistance: reward feedback, scenario presentation, and action advice, highlighting the different types of challenges faced by human and computer teachers under each category.

Reward Feedback. Perhaps the most direct way to engage a learning agent is to change the reward it receives from the environment. This approach is particularly well suited to human intervention when the environment does not offer any natural reward of its own. A major challenge is addressing the credit assignment problem: with domains that move quickly and have many states, human response times are not fast enough to assign their intended feedback to the correct state. Previous work, such as TAMER (Knox and Stone, 2009; Knox and Stone, 2010; Knox and Stone, 2012), addressed this by introducing a probabilistic error buffer for human reaction time to correctly attribute the signal to the intended state. Another problem is that people prefer to guide than to give feedback. In particular, (Thomaz and Breazeal, 2006) found that their human teachers tried to give guidance through the reward signal, which ended up confusing the agent.

Scenario Presentation. If the teacher has expertise both in the problem domain and in how machines learn, they can assist learning by presenting interesting states to the agent to help it learn to solve important sub-problems of the overall state space. In a previous work called Curriculum Learning (Narvekar et al., 2016), a computer teacher agent produces scenarios for a student agent to learn and transfer their knowledge to the larger domain. Their methods automatically develop scenarios while an agent is on a specific training trajectory, rather than do it before training begins. This form may be challenging for humans because knowledge of the underlying learning mechanism is essential when presenting the scenarios.

Action Advice. Another way to help an agent learn faster is to help it pick better actions at "teachable moments." This approach is challenging for both computer agent teachers and human teachers, but for different reasons. Computer teachers need a way to limit its influence so that the student does not just become a replica of the teacher; human teachers need a way to

intercept the learning agent before it makes its action selection.

Prior work on computer teachers limits their interactions with the students in a number of ways. First, there is an overall budget that restricts how much advice the teacher can provide the student throughout training (Torrey and Taylor, 2013). Second, some form of teacher-student interactions only occur during "important states," where the definition of an "important" state is chosen by the system designers, and could be modified if other definitions of importance appear. Prior work primarily differ in how provide the action advice in these "important states." For example, (Torrey and Taylor, 2013) proposed both *Advise Important*, where the computer teacher provides an action whenever the student's state is "important", and *Mistake Correcting*, which operates like *Advise Important* but has the additional constraint that the advice is provided only if the student would have chosen the wrong action. (Amir et al., 2016) expanded on Torrey and Taylor's work by creating a teaching framework titled apprenticeship learning. In apprenticeship learning, the student will actually query the teacher for advice when the student believes it is in an important state. Then, the teacher checks if the state is important (according to its model) and will provide the action if it truly is in an important state. This method performed similarly to the Torrey and Taylor strategies, however it required significantly less attention from the teacher, because as the student improved at its task, it relied less on the teacher. Ideally, once the student learned the proper action from the teacher in the important states, it didn't need to query the teacher anymore.

There has not been a great deal of prior work in which human teachers offer action advice, perhaps due to the difficulty in supporting the human teacher to provide timely action advice. One example we found is the work of (Maclin and Shavlik, 1996), where circumvent the issue by asking the human teachers to predict critical states and their corresponding correct actions *ahead of time*. In particular, they used a customized scripting language to provide the advice. The human, prior to training, would define states that they believed was important in the scripting language and then provide the correct action. This system worked well, but is difficult to scale to larger domains and include humans with no prior programming or RL experience.

3 TIME WARP HUMAN GUIDANCE INTERFACE

While prior work showed that pre-trained computer agents can successfully assist in training new agents, there are valid reasons to prefer human teachers. First, human teachers have a broader view of the problem than computer teachers and can naturally switch between the "big picture" and focusing on the specific details. Second, there are situations when a computer teacher is simply unavailable because there is no solution from a standard RL approach.[1]

3.1 Problem Formulation

We argue that human teachers may be best suited to offering *action advice* instead of providing a reinforcement signal or presenting scenarios. While reward feedback may be appropriate for tasks in which the environment does not naturally provide rewards on its own (Knox and Stone, 2012), as Thomaz found, humans naturally want to provide guidance rather than reward signal, resulting in a improperly shaped signal. While not yet thoroughly expored, we suspect that in order to craft highly useful scenarios, the human teacher must be experts in both the domain and RL methods.

In this paper, we propose a framework that facilitates assisted RL with action advice from a human teacher. This framework serves as a testbed that allows us to address two questions:

1. Can an RL agent assisted by action advice from human teachers produce a better performing policy than an RL agent with no assistance? Can our student agents converge on a policy in less time?
2. How do human teachers compare to computer teachers when both are giving action advice? For example, would the human teacher need to teach as often or as long as the computer teachers?

Inspired by the work of (Torrey and Taylor, 2013) and (Amir et al., 2016), our framework imposes similar constraints on our human teacher as their computer teachers. We record two main statistics from our learning agents. First, we record the total reward received by an agent in an exploitation run to determine the average episode reward for the agents at various points in training. We also record a statistic Amir used called *cumulative attention* to determine the amount of advice the teacher provided in a given period of time.

[1] Since a policy must have already been developed to create the computer teacher, creating a new student agent from scratch is arguably unnecessary.

3.2 Time Warp

Similar to the credit assignment problem that arises when a human has to provide the reward feedback, a major challenge toward facilitating action advice from human teachers is the need to address the differences in the perceived times granularity between the human and the student agent. To this end, we have developed the **Time Warp** interface, which provides the human teacher the ability to pause and reverse time while observing the student's learning progress.

The human teacher begins with a new untrained agent. The agent follows a standard RL algorithm and exploration function for action selection. In this work, we used Q-learning, with ε-greedy exploration. The underlying function approximation was quadratic, taking the form:

$$Q(s,a) = \sum_{i=0}^{n}\sum_{j=i}^{n} w_{i,j} f_i f_j + \sum_{k=0}^{n} w_k f_k, \qquad (1)$$

where where f_i are the features described in section 4.1, w_{ij} are the quadratic weights, and w_k are the linear weights. For all experiments we paused training every 100 episodes to record the average episode reward across 30 exploitation runs (where no exploration steps were taken and no updates were made to the Q-values).

While observing the agent under training, the human teacher decides whether and when to provide advice to the agent. This decision is based on the human's own perception of what an "important state" is; the human teacher is not informed about the state representation used by the computer agent.

To provide advice, the teacher activates a *take-control* signal. The system immediately pauses training and waits for further human input. The human could then cycle through the recently visited states in reverse chronological order. Once the human pinpoints the situation in which they want to advise, they would signal again, indicating that they will control the agent from this point on. The agent's world state would be restored to that same point, including the weights at that point in training. While the human controls the agent, any action given by the human would be used as the action input that updates the agent's weights according to the standard Q-learning update rule:

$$w_k \leftarrow w_k + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) f_k, \quad (2)$$

for the linear weights and,

$$w_{i,j} \leftarrow w_k + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) f_i f_j, \quad (3)$$

for the quadratic weights.

If the teacher provided no input actions, the agent would take no action, or NOOP. The teacher continues to control the agent until they either activate a *give-back-control* signal or guide the agent to a terminal state. If the human returned control, the agent would continue training where the human left off, following the default exploration function of the given RL algorithm.

Time Warp also limits the amount of advice the human is able to provide over the course of training. Both the advice budget (used also with the computer teachers) and the time limit restrict the human from falling simply into supervised learning or learning from demonstration. With Time Warp, we do not want infinite human interaction, but rather assistance during critical moments in training.

There are two important limitations on human teachers that distinguish them from computer ones. First, in real-time games, the possible action-rate is much higher than a typical human response rate, especially when a scenario requires reflection. In these cases, the learning agent might see multiple NOOPs as actions while the game state evolves around it. The ramification of this will be discussed below. Second, the humans can only participate in the training for comparatively brief stretches of time, as opposed to running constantly for days, as could be done with the computer agent. Whenever the teaching budget or the real time limit expires, the training of the agent will continue until convergence without a teacher present, using the same RL algorithm and exploration function that was used throughout human training that had no teacher.

4 THE TIME WARP FROGGER TESTBED

The testing domain that we selected to compare computer verses human teachers was the classic video game, Frogger. We chose it because decisions made by the agent had more long-term consequences compared to the other commonly used test domains, and it has been used by others working on similar problems (Griffith et al., 2013), (Subramanian et al., 2016). We compare agents that received no assistance, those that received action advice from computer teachers, and those that received action advice from non-expert human teachers using Time Warp.

Different human teachers may vary, not only in terms of their level of expertise at the domain and their computational literacy, but also in terms of their style of engagement with the learning agent. For example, some teachers may prefer to give frequent short bursts

Figure 1: Screenshot of Frogger with the various rows of cars, river objects, and randomly appearing flies and crocodiles in the home.

of advice micro-sessions while others might prefer to give fewer but longer sessions. To normalize across as much variability as possible, our experiment is carried out over a pool of 49 human teachers. In Section 4.3, we describe the measures we have taken to control for the variability in human teachers.

4.1 Frogger

Frogger is an arcade game originally developed by *Konami* in 1981. The goal of the game is to maneuver a Frog from the bottom of the screen to one of five homes at the top of the screen. To win the game, the player must place a Frog in each of the five homes. Between the starting location and the homes, the Frog must first avoid being hit by a number of cars while crossing a road and then cross a river by only hopping on floating objects in the river. Each row of either cars or river objects consists of varying number of objects all with varying speeds. When the Frog encounters the river section of the screen, it must not fall into the river by hopping on either logs or turtles. Some of the turtles will also dive, however for this work the diving turtles were disabled. There is also a random chance that a fly or crocodile will appear in the homes for a short amount of time. Flies provide more points for the Frog, while crocodiles kill the Frog. For this paper, we did not provide any additional points for capturing a fly in a home, but the agent would still die if it entered a home with a crocodile. Figure 1 provides a screenshot of Frogger.

With over 20 objects, 224 possible Frog locations, and five homes, the state space for Frogger is very large. Thus, we designed a feature set inspired from the work by (Emigh et al., 2016). One of their feature sets used a local view of the agent along with the y-location of the agent. The feature drew an imaginary 3x3 grid around the Frog, indicating if there was an object in each square with a 1 or 0 value. The y-location was then normalized in the range $[0, 1]$. We used two separate grids, one for road object and of for river objects. We built our Frogger implementation using an existing Frogger game built in Python, pyFrogger, and integrated it with the PyGame-Learning-Environment (padorange, 2013; Tasfi, 2016). A single life of the frog counted as one episode. Five possible actions were: up, down, left, right, and do nothing (NOOP). Unless blocked by the border of the screen, all five actions are always available.

4.2 Experimental Details

We used Q-Learning for the underlying RL algorithm with ϵ-greedy exploration when the human was not controlling the agent. The underlying algorithm used pessimistic initialization of the weights (setting the weights at 0 to start). Whenever the human was not providing an explicit action on the arrow keys, the system automatically selected the valid NOOP action for the human teacher. This meant when the human did nothing, the NOOP action was sent to the update rule for Q-Learning and it counted against the human's budget. Additionally, we limited the amount of states the human could reverse time. If the human provided the action signal within 50 states of a recent death, the human could reverse time up to 50 states before the last death. If the signal was given outside of 50 states from a death, the human could simply reverse time up to 50 states from the current state.

4.3 Human Teachers

For this study, we recruited 49 human teachers. All are college students from a variety of backgrounds. While some did have some knowledge of RL, most did not. Everyone was given a brief introduction: They were told that they would be observing an agent that was learning how to win the game of Frogger and that to learn this solution, it tried out many different actions and received positive and negative signals. We did not explain the reward structure of Frogger.

To match the computer teaching agents, we allotted each teacher a budget of 5,000 units of action advice but also a total time limit of 10 minutes for their advice session with their agent. Within this limit, teachers has the freedom to choose how they guide their agents: they can immediately take control of the agent; they can observe and only jump in when they see the agent making a mistake; they can opt to take

control multiple times; or they can have one continuous advice session until they deplete their budget.

Once the students completed training (by initiating the end themselves, running out of budget, or running out of time), we spawned 30 separate trials of standard Q-Learning with no advice to complete the training, in order to comapare to the computer teacher systems.

4.4 Baselines

We implemented three computer agent teaching strategies using the *best* set of weights from the standard Q-Learning run as the teacher. We implemented Torrey's **Advise Important** and **Mistake Correcting** strategies and Amir's **Ask Important-Correct Important** strategy in Frogger.

For all systems, we used the experimentally derived hyperparameters of $\alpha = 0.006$, $\gamma = 0.95$, and $\varepsilon = 0.04$. For the teaching parameters, Table 1 provides the best teaching parameters we found for each strategy through our experiments.

Metrics. In addition to recording the average episode reward for the computer teaching strategies, we also tracked the cumulative attention of the teacher, as developed by Amir This statistic measures the average number of states in which the teacher was asked to provide advice within a 100 episode window (across the 30 separate trials of each strategy). Each 100 episode frame's value was added to the previous 100 episode frame to track the statistic cumulatively. We slightly modified this statistic to measure the ratio of the number of states the teacher provided advice to the total number of states the agent observed. This was to allow us to directly compare experiments as some runs would observe more states in the 100 episode frames than others. We still tracked the ratio cumulatively, thus at any point in the plot, the value is the cumulative ratio of states the teacher provided advice in to the total number of states. If a plot flattens out, that means the budget ran out for the agent and no advice was provided after that particular episode.

5 EXPERIMENTAL RESULTS

As seen in figure 2, the human taught agents have similar performance to the three computer taught agents, and all four outperform the baseline average Q-Learning agent both in terms of the overall reward and the time taken to develop the policy.

Torrey's and Amir's strategies do scale well to the more complex domain of Frogger. Specifically,

Table 1: Teaching hyperparamters used for each system that we used to compare to our Time Warp human teaching system. The hyperparameters used for standard Q-Learning were kept constant throughout all systems (including the No Assistance system). These hyperparameters were: $\alpha = 0.006$, $\gamma = 0.95$, and $\varepsilon = 0.04$.

Teaching System	Advice Threshold	Budget
Advise Important	12.0	5000
Mistake Correcting	12.0	3000
Ask Important-Correct Important	10.0 (student), 12.0 (teacher)	5000
Time Warp	N/A	5000

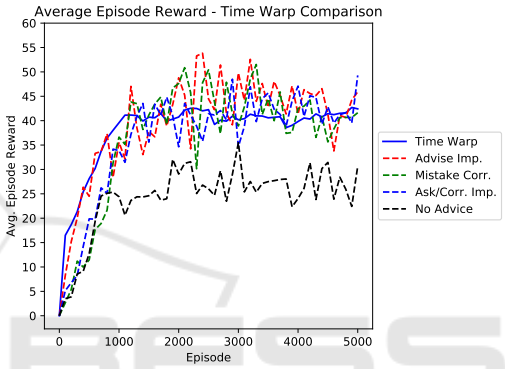

Figure 2: A comparison of the average episode reward for Q-Learning (no assistance), the three computer agent teachers, and Time Warp. As shown, Time Warp developed a similar performing policy to the computer agents in roughly the same amount of time. Regardless, all of the assistance systems converged on a better policy in less time than the no assistance agent converged policy.

Advise Important converged to an average episode reward of 44.80, **Mistake Correcting** converged to 42.46, and **Ask Important-Correct Important** converged to 41.78. **Time Warp** converged to 40.96 (averaged across trials of 49 human teachers). In comparison, a Q-Learning agent alone achieves an average reward of 26.38.

Note that Time Warp's curve has less variance than the other curves. This is because we have more data points for these curves. Each of the 49 human teachers help to train an eventual 30 separate agents, giving a total of 1,470 agents produced from Time Warp. Conversely, the computer agent systems only had 30 different agents to average.

Attention Level from Teachers. An ancillary goal of teacher-based systems is to minimize the amount of attention provided by the teachers during the teaching process. Figure 3 compares the cumulative frac-

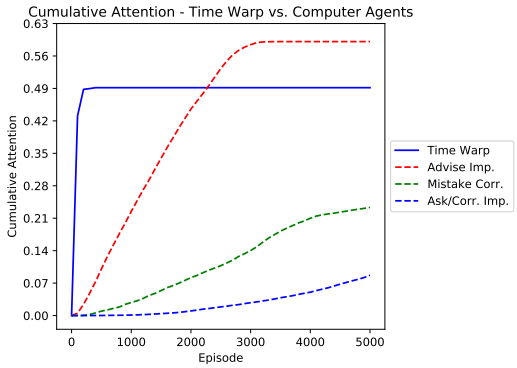

Figure 3: A plot of the cumulative attention of the three computer based teaching strategies and Time Warp. Agents under Time Warp accumulated attention only within the first 100 episodes of training because most of the humans used up their budget quickly. This equates to less than 10 minutes of the teacher's time. In contrast, computer teachers must stay with the agents throughout the entirety of their training.

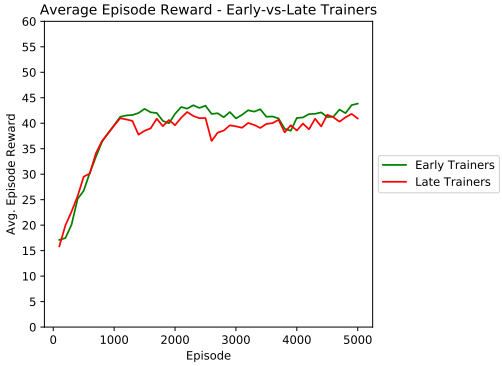

Figure 4: A comparison of agents trained by humans who used up their advice budget early and those trained by humans whose finished advising later. The final difference is not statistically significant.

tional attention of the three computer guidance systems and Time Warp. While agents learning under Time Warp received more cumulative attention than **Mistake Correcting** and **Ask Important-Correct Important**, the attention was concentrated in the beginning of training, since the human teachers do not stay with the learning agents throughout their full training.

In general we find that this statistic does not truly take into account how involved the teacher was in training. For Time Warp, our human trainers provided advice for no more than 10 minutes of real training time. The computer teaching systems require the teacher to be present for the entirety of training. Therefore, the computer teaching systems require the teacher to provide much more attention in terms of total time spent observing the agent training. While they may not provide advice in as many states (and as quickly) as the Time Warp system, they do need to be present for the entire training phase of the agent. For our purposes, this is a best case scenario for a human teaching system. Humans do not have the patience to watch an agent converge on a policy, so the interaction must be done in the short time span that the human can fully concentrate. This also means that the advice the human provides in this short time frame is meaningful and insightful to the agent.

5.1 Discussions

A number of interesting questions arose in the context of providing human action advice. Does the amount of time the human teacher spent training their agent affect the overall performance of the agent? How is the student learning affected by the quality of the teachers? Does the student agent perform better immediately following the human advice session than after further self-training? To begin to answer these questions, we analyzed our collected data to look for differences between different groups of teachers.

First, we investigated whether the amount of time the human spent training affected the performance of the agent. We wanted to see if human teachers that left training late performed better than the human teachers that left training early. We separated early and late trainers based on the average stopping point (episode 96). If a teacher quit training before episode 96, they were labeled an early trainer and trainers that quit after episode 96 were labeled late trainers. We then averaged across these two groups to produce the plot in Figure 4. Counter to our intuition, the difference between these two groups was not significant (the early trainers had a higher average of 41.89, while the late trainers had an average of 39.98).

Second, we explored the difference between "good" and "bad" teachers. We defined "good" teachers as those with the better performing agents the end of human training. Since they were able to help their agents improve quickly, they might have had some good advising strategies. We wanted to see whether these students of "good" teachers would continue to outperform their peers as they learn on their own.

To make the group assignments, we ran 30 exploitation runs with the weights developed by the agent at the end of human training to compute the average reward for the population. We consider any agent with a reward greater than the average (11.70) as having had a "good" teacher; analogously, the other agents are said to have had a "bad" teacher. Figure 5 shows the reward averages for the two groups as the agents continue to train on their own.

Counter to our expectation, the agents learning

Figure 5: A comparison of agents trained by "good" and "bad" teachers (defined by agent performance at the end of human advising sessions). As the agents continued training on their own, the "bad" students' performance overtook the "good" students.

from "bad" teachers had a better overall final average (42.22) than those learning from "good" teachers (39.27). The difference is statistically different with $p = 0.006$. A possible explanation is that the "bad" teachers might have encouraged more exploration of the entire space than exploiting the reward structure of Frogger. Once the standard Q-Learning with ε-greedy exploration algorithm took over, the agents focused more on reinforcing the weights that led to high reward, and eventually learned a better policy.

6 FUTURE WORK

There are many areas of potential future work to pursue, three direct extensions, and one alternative method. First, we would like to determine the effects of varying the expertise of the agent when the human teacher enters training. Does a human teacher add more benefit to a completely untrained agent, or to an agent that has some knowledge of the space?

Second, we would like to investigate the difference between small and large training budgets (both in actual time to train or number of advised actions).

Third, we would like to determine the extent of the impact of variations in the expertise of the teacher. In particular, looking at both experts in terms of playing Frogger verses players new to the game, and experts in RL, verses those unaware of how it works.

Finally, we are developing a mechanism for allowing a human trainer to develop interesting problems for the agent to learn how to solve (scenario presentation). We expect human trainers will be able to provide meaningful scenarios that can be more beneficial that automatically generated scenarios.

7 CONCLUSION

This work showed that it is possible to integrated a human teacher into the training of an RL agent by providing a leveling platform to accommodate a human's slower response time and potentially different mental model of the state space. The Time Warp guidance interface that we developed can facilitate the interaction between a human teacher and a learning RL agent by giving the human teacher the ability to reverse time to an important state where the human could then provide a sequence of actions for the agent to take. This mechanism addresses the credit assignment problem and allowed the human to direct the exploration and guide the agent toward the areas in the state space of high reward in order to converge on a better-performing policy faster.

Validated by experiments on the Frogger test domain, we compared human action advising via Time Warp with different baselines. Our results show that:

- Time Warp converged on a better performing policy in less time than a Q-Learning agent with no assistance
- Time Warp converged on a similar performing policy in similar time as compared to the computer agent teaching strategies
- While the human teachers provided more advice (as measured by cumulative attention) than the computer teachers, the human teachers interacted with their agents for no more than 10 minutes, whereas the computer systems required a teacher present throughout the entirety of training
- How long the human trainers spent with their agents did not significantly change the policy of their agent, but the teachers that produced worse performing agents at the conclusion of human training had a significantly better policy after finishing training with a standard RL algorithm most likely due to increased exploration of the space

We conclude that, with some appropriate mediation, humans can be effective teachers for learning agents. Humans are skilled at knowing the right action in specific states (in terms of immediate reward) as well as the ability to keep the larger picture in mind, essential to proper planning within a complex environment. Humans are able to determine the future expected reward in an environment quicker (and without as much random exploration) than a computer agent. With this ability, a human can better direct and guide an agent toward the development of a policy and properly direct the right exploration through the space.

REFERENCES

Amir, O., Kamar, E., Kolobov, A., and Grosz, B. (2016). Interactive teaching strategies for agent training. In *International Joint Conference on Artificial Intelligence*, New York City, USA.

Emigh, M., Kriminger, E., Brockmeier, A., Príncipe, J., and Pardalos, P. (2016). Reinforcement learning in video games using nearest neighbor interpolation and metric learning. *IEEE Transactions on Computational Intelligence and AI in Games*, 8:56–66.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pages 2625–2633.

Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The TAMER framework. In *The Fifth International Conference on Knowledge Capture*.

Knox, W. B. and Stone, P. (2010). Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*.

Knox, W. B. and Stone, P. (2012). Reinforcement learning from simultaneous human and MDP reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Maclin, R. and Shavlik, J. W. (1996). Creating advice-taking reinforcement learners. *Machine Learning*, 22(1):251–281.

Narvekar, S., Sinapov, J., Leonetti, M., and Stone, P. (2016). Source task creation for curriculum learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, Singapore.

padorange (2013). pyfrogger. https://sourceforge.net/projects/pyfrogger/, accessed November, 2020. Also available at https://www.usna.edu/Users/cs/crabbe/pyFrogger0.1.zip.

Subramanian, K., Isbell Jr, C. L., and Thomaz, A. L. (2016). Exploration from demonstration for interactive reinforcement learning. In *Aamas*, pages 447–456.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition.

Tasfi, N. (2016). Pygame learning environment. https://github.com/ntasfi/PyGame-Learning-Environment, accessed November, 2020. Also available at https://github.com/profcrabbe/PyGame-Learning-Environment.

Thomaz, A. L. and Breazeal, C. (2006). Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 1000–1005. AAAI Press.

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*, pages 1053–1060.