Deep Implicit Imitation Reinforcement Learning in Heterogeneous Action Settings

Iason Chrysomallis, Georgios Chalkiadakis, Ioannis Papamichail, Markos Papageorgiou

Technical University of Crete, Chania, Greece
ichrysomallis@tuc.gr, gchalkiadakis@tuc.gr, ipapamichail@tuc.gr, mpapageorgiou@tuc.gr

Abstract

Implicit imitation reinforcement learning (IIRL) is a framework that aims to aid a *trainee* agent's learning process via observing the state transitions of a *mentor*, but without access to the latter's action information. Standard IIRL assumes a shared Markov decision process (MDP) between the mentor and trainee, consequently implying an identical action space. This restriction imposes limitations on the applicability of implicit imitation frameworks in real-life scenarios where, possibly due to variations in physical characteristics, the mentor agent may possess distinct own actions, thereby creating a heterogeneous action setting. In this work, we extend the deep implicit imitation Q-networks (DIIQN) method—an online, model-free, deep RL algorithm for implicit imitation—to allow for heterogeneous action sets between mentor and trainee agents. Equipped with our *heterogeneous actions DIIQN (HA-DIIQN)* method, a trainee agent can harvest the benefits of IIRL even in heterogeneous action settings, achieving accelerated learning and outperforming non-optimal mentor agents.

Introduction

As the complexity of environments tackled by deep reinforcement learning (DRL) algorithms escalates, the demand for accelerated learning methodologies arises (Arulkumaran et al. 2017; Hussein et al. 2017). Given this, *imitation-based RL* algorithms (Price and Boutilier 2003; Abbeel and Ng 2004; Zhang et al. 2021; Da Silva et al. 2020; Wang, Warnell, and Stone 2023) have received attention primarily due to their capacity to expedite the learning process. Guided by an expert agent, the *mentor*, a *trainee*, or *observer*, agent can speed up its training, thereby directing its efforts towards optimizing its behavior. While numerous methodologies exist for *explicit imitation* (Hussein et al. 2017; Argall et al. 2009), where complete mentor information is accessible (including state transitions, actions taken, and occasionally, rewards received), constraints such as communication barriers or other limitations may render the utilization of these methodologies unfeasible. *Implicit imitation RL* (Price and Boutilier 2003) circumvents these obstacles by solely requiring state transition observations from the mentor, thus facilitating faster convergence.

In this context, *deep implicit imitation Q-networks (DIIQN)* (Chrysomallis et al. 2023) is a recently proposed online, model-free, DQN-based (Mnih et al. 2015) technique. Operating with solely the state transition data from the mentor, the DIIQN algorithm employs state similarity assessment and internal exploration mechanisms to infer the mentor's actions, computing augmented loss functions for neural network parameterization. DIIQN diverges from conventional behavior cloning methodologies (Torabi, Warnell, and Stone 2018) by not being bound by the performance limitations of the mentor, thus allowing for potential performance enhancement beyond that of a sub-optimal mentor.

Nevertheless, like virtually all other imitation approaches,[1] the DIIQN framework operates on the premise of a homogeneous action setting—a shared, uniform action environment among agents, wherein actions executed by the mentor are equally accessible to the learner. However, depending on their physical characteristics, agents may have different action capabilities due to inherent differences in their physical and cognitive abilities. For instance, variations in the size of robot components can significantly influence the outcome of actions, yielding disparate results. In this paper, we challenge the assumption of homogeneous action sets, by proposing a methodology designed to offer advantages to the training agent even in scenarios where its action set differs from that of the mentor. Leveraging fundamental mechanics of the DIIQN algorithm, we initially examine whether or not a mentor state transition of interest is feasible by the trainee. Subsequently, in infeasible cases, we endeavor to identify potential alternative pathways—intuitively, building "bridges" to overpass the inaccessible mentor trajectory sections—for further investigation. The corresponding feasibility-related information obtained is transferred to the updated augmented loss functions to suitably adjust the neural network parameters.

In summary, our main contributions are the following. We provide HA-DIIQN, a novel deep IIRL method for heterogeneous action settings, retaining the DIIQN's online and model-free characteristics; while it breaks free of specific rigid assumptions previously prevalent in the literature regarding heterogeneity (Price and Boutilier 2001). We provide an indicator for action feasibility in the presence of het-

[1] With the notable exception of (Price and Boutilier 2001).

erogeneity, as well as a mechanism to identify state transition bridges. Moreover, we provide the necessary augmented loss functions, refining those of DIIQN, to integrate this required information. Our experimental results confirm that the benefits associated with deep implicit imitation, namely accelerated training and improved performance over the mentor agent, are achieved even in contexts with non-homogeneous action settings. Our experiments are conducted both in a maze environment similar to those used for the study of tabular implicit imitation RL (Price and Boutilier 1999, 2001, 2003), and in a challenging force-actuated navigation environment (Fu et al. 2020).

Background and Related Work

Here we provide background and review related work.

Deep Q-Networks

Deep Q-Networks (DQN) (Mnih et al. 2015) blends Q-learning with neural networks to tackle complex decision-making problems with high dimensional state spaces. In its core, DQN employs a neural network, namely the Q-network, that approximates the Q action-value function, enabling the handling of complex state representations and non-linear decision boundaries. At each time step t, starting at state s and taking the action a, the agent reaches the next state s' and receives a reward r, creating a transition tuple $\langle s, a, r, s' \rangle$. With network weight parameters θ_t, the Q-network representation $Q(s, a|\theta_t)$ provides us with an estimation of said action. Additionally, DQN employs the target network, a neural network snapshot of the Q-network with regularly updated parameters θ_t^-, to stabilize the training process. Setting a discount factor γ, the algorithm updates the network's weights in accordance with the loss function:

$$L(\theta_t) = E_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s', a'; \theta_t^-) - Q(s, a; \theta_t))^2] \tag{1}$$

Experience replay (Lin 1992; Mnih et al. 2015) is central to its success. It is a mechanism dedicated on storing past agent experiences and randomly sampling them to alleviate sample correlations during training, improving efficiency and stabilizing learning. From a data structure's point of view, experience replay is a sizeable *queue* that stores subsequent, fully observable transitions encountered by the agent.

Deep Implicit Imitation Q-Networks

The DIIQN algorithm (Chrysomallis et al. 2023) is based on the principles of DQN, but designed to enable deep implicit imitation RL with homogeneous actions. Implicit imitation involves two distinct agent roles (Price and Boutilier 2003): the mentor and the observer. The observer, which is the agent undergoing training, learns through conventional interactions with the environment while benefiting from mentor guidance evaluated at a proficient level. This guidance is conveyed solely through state transitions, refraining from communicating a complete set of information (including actions undertaken and rewards received) as studied in explicit imitation techniques (Silver et al. 2016; Nair et al. 2017; Vecerik et al. 2017; Hester et al. 2018; Kang, Jie, and Feng 2018). Consequently, the observer is tasked with estimating the absent corresponding actions and approximating the mentor's behavior. The main objective of implicit imitation learning is to accelerate the observer's learning process. Notably, implicit imitation doesn't aim to clone the mentor's behavior but leverages it for faster learning, potentially exceeding the mentor in non-optimal cases.

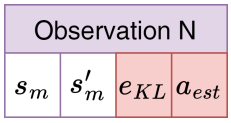


Figure 1: Imported mentor observation, highlighted with its estimated error and action

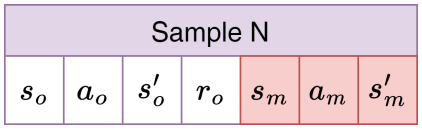


Figure 2: Augmented experience tuple

DIIQN incorporates this methodology via the following protocol. Initially, it acquires a mentor dataset comprising state transitions $\langle s_m, s'_m \rangle$. To effectively utilize these transitions, an action estimation process is undertaken for each transition. To this end, two new attributes are appended to each mentor transition (Fig. 1): the estimated action a_{est} and a "running error" estimate, e_{KL}, that uses Kullback-Leibler (KL) divergence for multivariate normal distribution over the state features (Duchi 2016) as a metric to denote the disparity between this estimated action and the ground truth. At each iteration, a comparison is conducted between the current observer state transition $\langle s_o, s'_o \rangle$ and all transition tuples $\langle s_m, s'_m \rangle$ contained within the mentor dataset. If the calculated e_{KL} error falls below the previously assigned one, it is implied that the action a_o taken by the observer more accurately reflects the mentor transition. Consequently, an update is performed on both the estimated action a_{est} and the associated error e_{KL}. This process can be seen as one filling in "missing information" (the action taken by the mentor).

The subsequent procedure unfolds as follows: At each time step, a similar mentor state s_m is selected. This selection process (Fig. 3) entails identifying the most similar one and executing a Nearest Neighbour search (Goldberger et al. 2004) on the mentor's dataset restricted by an empirically set threshold e_{th}^{KL}. From this set, a random mentor initial state, s_m, is chosen, and its corresponding subsequent state, s'_m, along with the estimated action, $a_m = a_{est}$, are utilized to create an augmented experience tuple $\langle s_o, a_o, s'_o, r_o, s_m, a_m, s'_m \rangle$ (Fig. 2). This ensures variability on selected mentor transitions, increasing model stability.

Upon the completion of populating the augmented experience tuple, the observer effectively has access to two distinct perspectives that can be employed to build a loss function: one taking into account solely its own experiences, and one taking into account the *estimated* mentor actions. It employs these to build two distinct loss functions, the "observer loss

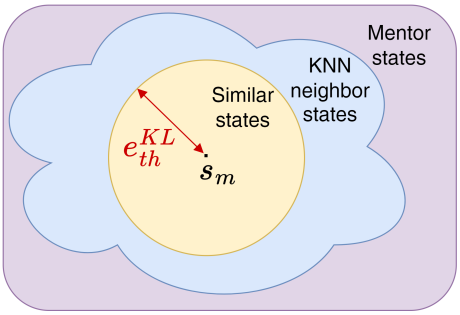


Figure 3: State similarity visualization

function" L_o, and the "mentor loss function" L_m (though this is still constructed and used by the observer):

$$L(\theta_t)_o = E_{s,a,r,s'}[(r_o + \gamma Q(s'_o, \underset{a'}{\operatorname{argmax}}\, Q(s'_o, a'; \theta_t^-); \theta_t^-) - Q(s_o, a_o; \theta_t))^2] \quad (2)$$

$$L(\theta_t)_m = E_{s,a,r,s'}[(r_o + \gamma Q(s'_m, \underset{a'}{\operatorname{argmax}}\, Q(s'_m, a'; \theta_t^-); \theta_t^-) - Q(s_m, a_m; \theta_t))^2] \quad (3)$$

However, observer model parameterization requires only one loss function per sample. Its determination is made given past Q values for both the observer and the (estimated) mentor actions. At each step, a Q_o^{past} and a Q_m^{past} value is stored and two square distances are calculated:

$$D_o = ||Q(s_o, a_o; \theta_t) - Q_o^{past}||^2 \quad (4)$$

$$D_m = ||Q(s_m, a_m; \theta_t) - Q_m^{past}||^2 \quad (5)$$

and $\min\{D_o, D_m\}$ is selected for network optimization.

Related Work

A plethora of works have examined heterogeneity in transfer learning, albeit not in the context of disparate action settings. These studies primarily delve into the heterogeneity present in various state spaces and their characteristics (Day and Khoshgoftaar 2017; Bao et al. 2023). When the mentor and observer exhibit dissimilar observable features, incorporating feature transformations to facilitate knowledge transfer is required to address this divergence. More importantly, these works are further categorized into supervised (Zhao and Hoi 2010; Yan et al. 2016; Duan, Xu, and Tsang 2012), semi-supervised (Wang and Mahadevan 2011; Tsai, Yeh, and Wang 2016; Harel and Mannor 2010), and unsupervised learning (Yeh, Huang, and Wang 2014; Zhou et al. 2014; Prettenhofer and Stein 2010) paradigms, based on the availability of labeled data, yet lacking RL implementations.

The sole investigation into action heterogeneity settings in implicit imitation reinforcement learning is attributed to (Price and Boutilier 2003, 2001). Their work focuses on *tabular*—non-deep—*model-based* reinforcement learning. They introduce an action feasibility test, built upon model-based principles, responsible to identify instances where a mentor transition is deemed infeasible due to action capabilities mismatch between the mentor and observer. Afterwards, through simulated steps, they endeavor to workaround infeasible actions by parsing through a k depth exhaustive search. They demonstrate the observer's ability to effectively leverage the mentor's guidance, even within environments characterized by heterogeneous action settings, thereby circumventing unnecessary confusion or being misled. Their work serves as a fundamental influence for ours, though we operate within a *model-free* and *deep RL* framework. Moreover, in contrast to their methodology, our approach does not rely on the assumption of simulating environment steps or employing brute force exploration to uncover potential bridges to the mentor's trajectory path.

Heterogeneous Actions DIIQN

In this section we present in detail HA-DIIQN, the first method for deep implicit imitation RL in heterogeneous action settings. HA-DIIQN takes inspiration from the tabular implicit imitation RL method of (Price and Boutilier 2001, 2003), notably removing important assumptions present in that work—specifically, having the ability to simulate steps or have information regarding high-value states before exploring them. Moreover, as previously mentioned, HA-DIIQN builds on DIIQN (Chrysomallis et al. 2023), broadening its functionality to handle actions heterogeneity.

Specifically, we introduce a mechanism aimed at detecting potential state transition infeasibility, caused from action heterogeneity within a mentor transition. In instances where such discrepancies are identified, we employ a search operation utilizing the experience replay component to explore potential alternative paths intersecting with the mentor's trajectory. If a viable alternative path is discovered, we utilize its parameters—instead of the mentor's infeasible ones—to populate the augmented experience replay sample. To this end, we propose novel, distinct to those of DIIQN, augmented loss functions for use by HA-DIIQN; these are introduced so as to suit the augmented experience replay tuple constructed given the identified action infeasibility.

Identifying Transition Infeasibility

Determining the feasibility of an observed state transition in the presence of heterogeneous action settings poses a challenge. Despite the observer possessing comprehensive information regarding their action space, similar information is not available from the mentor in an implicit imitation scenario. Hence, when selecting a mentor transition $\langle s_m, a_m, s'_m \rangle$ for training, where a_m denotes the mentor estimated action by the observer, it becomes necessary to employ a mechanism to determine its feasibility, and subsequently refraining from incorporating potentially misleading information. In many instances, reliance on such information may confuse the observer into attempting to learn from a policy that is not feasible within their operational context.

At each iteration, the observer selects a mentor transition (Fig. 1). The mentor action a_m is derived from the observer's own exploration, while the associated error e_{KL} represents

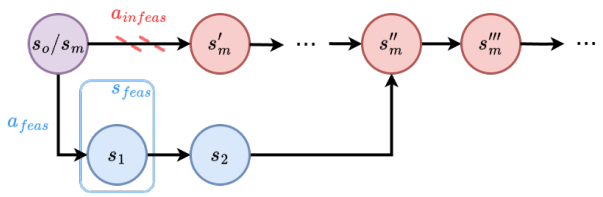


Figure 4: Bridge discovery. Starting from the observer state s_o with a similar mentor state s_m ($s_o \approx s_m$), the mentor transition to s'_m, using action a_{infeas}, is infeasible to the observer. However, there is a bridge available to them via $s_o \rightarrow s_1 = s_{feas} \rightarrow s_2$, using action a_{feas}, that interjects with mentor trajectory $s_m \rightarrow s'_m \rightarrow ... \rightarrow s''_m \rightarrow s'''_m$ at s''_m.

how far the observer transition with this action was from the mentor transition during exploration. In essence, this error can also be interpreted as a feasibility indicator of the mentor transition given the estimated mentor action. A high error means that either the transition is indeed infeasible and should be labeled as such; or that it is feasible, but the observer's estimation of the mentor action is inadequate.[2]

In both cases, it is evident that mentor information is unlikely to prove beneficial to the observer. Consequently, we establish criterion for potential infeasibility, utilizing as metric the error e_{KL} associated with each mentor transition. Here, we introduce an empirically defined hyperparameter e_{infeas}, to gate such transitions. If the error e_{KL} falls below the threshold set by e_{infeas} ($e_{KL} < e_{infeas}$), the transition is deemed feasible, and normal training procedures for the DIIQN algorithm ensue. Otherwise, the transition deemed as infeasible is appropriately flagged for subsequent handling.

Handling Transition Infeasibility (via "Bridging")

Following the identification of a mentor transition as infeasible, one option is to deactivate the mechanisms reliant on the mentor's perspective and proceed accordingly. However, an alternative approach involves devising a method to capitalize on such transitions. Despite the observer's inability to reach the next mentor state s'_m at the current time step, it remains plausible that a transition to that state could be attained within subsequent steps. It is also conceivable that within those the observer may not reach s'_m, but rather a subsequent mentor state along their trajectory (Fig. 4).

It becomes evident that a mechanism is required to bridge the gap between the observer's current position s_o, and the trajectory of mentor states commencing from s_m. To address this need, we introduce a bridging mechanism which we term the *deep k-n step repair* (Alg. 1). The primary objective of this mechanism is to identify alternative paths, hereafter referred to as $bridges$, allowing the observer to intersect the mentor trajectory. This mechanism relies on two parameters: the maximum search depth for the observer, denoted as k, and the maximum search depth for the mentor, n.

[2]We should note here that, in the second scenario, the observer may, in subsequent iterations, discover a more appropriate estimated action with a sufficiently reduced error, thus meeting the criteria for feasibility. This progression is inherent to DIIQN.

Algorithm 1: Deep k-n step repair

Input: Observer initial state s_o, Observer experience replay R, Observer maximum depth k, Mentor initial state s_m, Mentor extracted demonstrations D, Mentor maximum depth n
Output: Bridge $trajectory$

1: **for** $depth_m$ in range $1...n$ **do**
2: s_m^{future} = findMentorState($s_m, depth_m, D$)
3: $trajectory = \{\}$
4: **for** $depth_o$ in range $1...k$ **do**
5: $s_o^{future}, a_o^{future}$ = findObserverTransition($s_o, depth_o, R$)
6: $trajectory = trajectory \cup (s_o^{future}, a_o^{future})$
7: **if** areSimilar($s_o^{future}, s_m^{future}$) **then**
8: **return** $trajectory$
9: **end if**
10: **end for**
11: **end for**
12: **return** $\{\}$

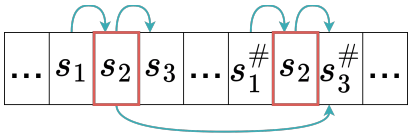


Figure 5: Cross reference in experience replay. We begin the search from a state s_1 similar to the initial observer state s_o. The initial path from this state is represented by the sequence $s_1 \rightarrow s_2 \rightarrow s_3$. Although $s_1^{\#}$ is not similar to s_o and search would typically skip its trajectory, cross referencing s_2 enables the exploration of the additional path $s_1 \rightarrow s_2 \rightarrow s_3^{\#}$.

The bridging mechanism involves examining up to n mentor transitions (l. 1-2 in Alg. 1) and up to k observer transitions (l. 4-5 in Alg. 1), whilst keeping track of the observer explored path (l. 6 in Alg. 1). As this explored path is derived from the observer's replay memory, we possess complete information about the transition, including both the subsequent states and the actions taken to reach them, which are temporarily stored in the trajectory. At each corresponding iteration, if the two states are deemed similar (l. 7 in Alg. 1, executed as discussed in (Chrysomallis et al. 2023)), it is concluded that a bridge has been identified, thereby providing the complete path (l. 8 in Alg. 1). The selection of the k and n values is domain-specific, but as a rule of thumb it should be restricted to low values to control the required computational power for operation.

The methodology for accessing subsequent states (l. 2 and 5 in Alg. 1) varies between the mentor and the observer. For the observer, we initiate a search within the experience replay populated with previously explored transitions. We begin by finding similar states to s_o. From that point, since the experience replay operates as a queue, its inherent sequential nature allows us to traverse their subsequent entries sequentially. Additionally, we employ a method of finding parallel paths, where states along the current path are "cross-

referenced" to identify alternative paths with different preceding states (Fig. 5). As the experience replay consists of transitions initiated by the observer, it is clear that the effectiveness of this strategy is dependent on the exploration of the observer. Given that experience replay is assigned a maximum size, newly generated entries may overwrite the oldest transitions, thereby eliminating potential identified bridges. To address this limitation, three additional variables are appended to each mentor transition: the first action a_{feas} and next state s_{feas} of the trajectory bridge (Fig. 4) and length of the bridge l_{feas} This information is updated whenever a shorter bridge is found, as our objective is to identify only the shortest possible path into the mentor's trajectory. Storing all possible bridges does not provide any additional value, as no new information is gained from doing so. In this way, if previously identified bridges become inaccessible due to deletion from the replay memory, they can still be retrieved externally of the replay memory.

By contrast, accessing subsequent states for the mentor is straightforward, as the provided dataset is inherently ordered. Thus, we can proceed from the initial mentor state s_m. If the dataset is not in sequential order, a mechanism similar to that utilized by the observer may be employed.

Training with Infeasibility

We now exploit our bridging mechanism in order to appropriately train our neural network. If a "bridge" to future mentor states was not found, we do not allow the use of the L_m (Eq. 3) loss function. In this scenario, no information from the mentor is incorporated to prevent potentially misleading the observer. However, in the event that a bridge is identified, adjustments are made to the mentor's loss function L_m. Instead of using the next mentor state s'_m and the estimated mentor action a_m, we employ the first action a_{feas} and next state s_{feas} of the bridge (Fig. 4):

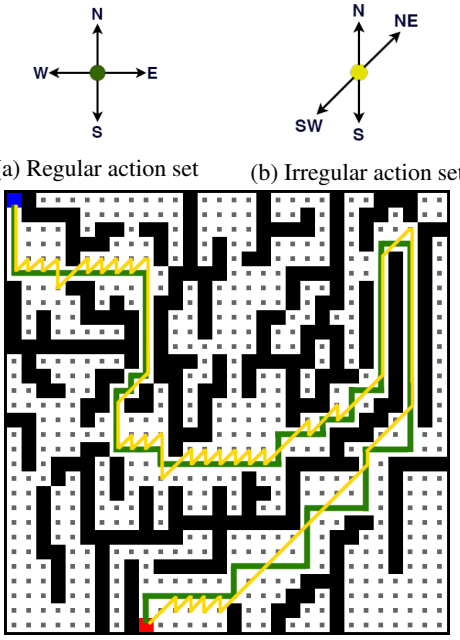
$$L(\theta_t)_m = E_{s,a,r,s'}[(r_o + \gamma Q(s_{feas}, \underset{a'}{\operatorname{argmax}}\, Q(s_{feas}, a'; \theta_t^-); \theta_t^-) - Q(s_m, a_{feas}; \theta_t))^2] \tag{6}$$

The reason we do not utilize the entire bridge trajectory is to retain the off-policy property inherent in DQN methods. Even though the selected action a_{feas} does not directly lead the observer to the mentor-suggested trajectory, it nonetheless provides high value information on how to access it.

Experimental Evaluation

We now present a systematic assessment of our algorithm on two distinct environments: namely, a *2D maze* environment (Brockman et al. 2016) and a more complex, force-actuated navigation *Point Maze* setting (Fu et al. 2020)[3].

[3]In the technical appendix (https://osf.io/tkqp2/), we provide additional experiments in the challenging *autonomous driving in lane-free traffic* environment, used also in the paper introducing DIIQN (Chrysomallis et al. 2023). We note that this setting is *not* a natural heterogeneous actions one. Regardless, HA-DIIQN still provides an edge in performance even in that setting. In some detail, given that the mentor operates within a continuous action space while the observer within a discretized version of that space, certain transitions feasible for the mentor will be unattainable for the observer. While these discrepancies remain undetectable in the DIIQN framework, HA-DIIQN is able to identify them and to approximate a more accurate discretization of the continuous action space via its bridging mechanism—leading to enhanced performance.



(a) Regular action set (b) Irregular action set

(c) Visualization of 2D Maze 30×30 with optimal paths for regular action set (green) and irregular action set (yellow).

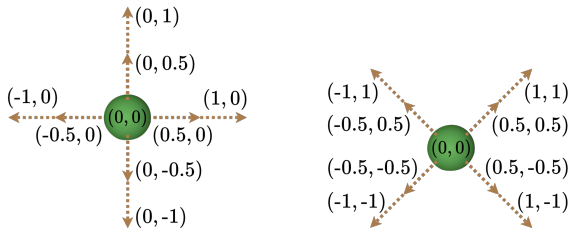
Figure 6: 2D Maze

Experimental Setup

Maze 2D To demonstrate the benefits of our approach, we first test in a 2D maze environment. Though simple to comprehend, a maze environment allows for the thorough testing of our action heterogeneity-tackling method, as was the case for the tabular heterogeity-aware IIRL method of (Price and Boutilier 2001, 2003) (tested in smaller—10×10 sized—mazes than those we use here).

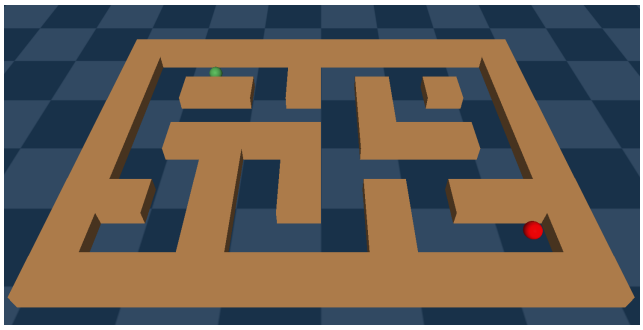
Specifically, we utilize the 2D Maze environment from OpenAI Gym (Brockman et al. 2016), introducing a maze of size 30×30 where the state space comprises the agent's current position. The reward function is designed to penalize collisions with walls or boundaries and revisiting states, offering a small negative reward for each step and a significant positive reward upon reaching the goal.

In the context of the HA-DIIQN algorithm, we introduce two distinct action sets: one orthogonal, denoted as *regular* (Fig. 6a), and another designated as *irregular* (Fig. 6b). These action sets generate different optimal paths for our maze configuration (Fig. 6c). Notably, the mentor agent navigates according to the regular action set, while the observer employs the irregular action set, thus highlighting the heterogeneous nature of the environment.



(a) Orthogonal action set with exerted linear forces (in N)

(b) Diagonal action set with exerted linear forces (in newtons N)



(c) Large Point Maze map with positioned agent (green) and target goal (red).

Figure 7: Point Maze (Fu et al. 2020)

Progress tracking is conducted based on episodes, defined by either reaching the goal or exceeding 4000 steps without reaching it, at which point the environment resets. The environment is considered solved when the agent achieves a reward per episode of -3.4 with the irregular action set.

We stress that despite its seeming simplicity, a 2D maze environment poses significant challenges when applied to large dimensions, such as the 30×30 configuration chosen in our work here. We selected this environment to showcase performance in a straightforward setting with clear action distinctions between the mentor and observer agents.

Point Maze To further illustrate the strengths of our method, we also utilize a continuous space environment. Specifically, we employ D4RL's Point Maze (Fu et al. 2020), an environment based on the MuJoCo simulation physics engine (Todorov, Erez, and Tassa 2012), introducing a force-actuated ball agent tasked with reaching a designated target goal. The presence of inertia in the environment introduces additional complexity, rendering the problem more challenging to solve due to the continuous and momentum-dependent nature of the agent's movements, which complicates precise control and decision-making.

In this simulation environment, the continuous state space includes the agent's longitudinal and lateral position x, y, and the velocities along these axes v_x, v_y. The reward function provides positive reinforcement only for reaching the goal (Fu et al. 2020), while imposing a small penalty (-0.001 per step) to discourage unnecessarily long paths.

Additionally, we have two distinct action spaces, both discretized: an *orthogonal* action space (Fig. 7a) utilized by the mentor, and a *diagonal* action space (Fig. 7b) employed by the observer. Each action set comprises $8 + 1$(no-op) actions, with each action corresponding to a force applied, measured in newtons (N). The action space available to the observer is entirely distinct from that of the mentor, with no overlap between the two except for the no-action option.

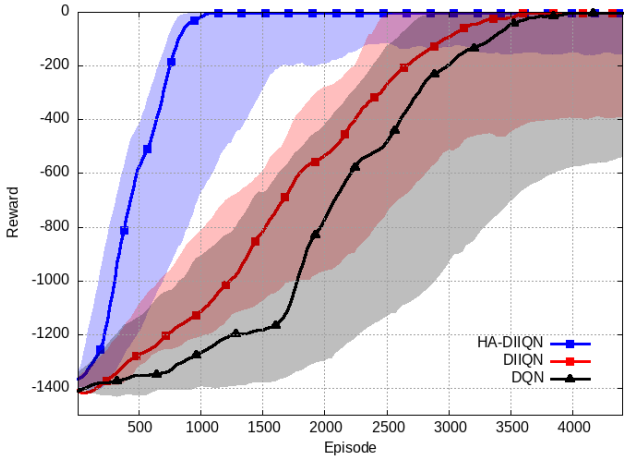


Figure 8: 2D Maze: reward per episode comparison, averaged over 30 runs. Shaded area depicts standard deviation.

Each episode consists of 4000 steps, and, based on the selected maze setup, an episode is considered solved when it consistently surpasses the lower bound of -0.6 reward per episode. Due to the inertia inherent in the force-actuated mechanics of the simulation, minor fluctuations in reward are expected, even as the system approaches convergence.

Experiments and Results

Maze 2D For the experimentation, a singular optimal path is used as the mentor trajectory demonstration, as illustrated in Figure 6c, employing the regular action set (Fig. 6a). All showcased agents are navigating according to the irregular action set (Fig. 6b), immediately introducing heterogeneity into the setup upon the application of imitation techniques.

In Fig. 8 we present the performance of three distinct agents measured by their collected rewards over 4500 total number of episodes. Initially, we showcase the performance of a baseline DQN agent, lacking any mentorship or imitation module, serving as a reference point for comparative analysis. Subsequently, we examine the accelerated training exhibited by our proposed HA-DIIQN agent, which utilizes the heterogeneous mentor's dataset, showcasing its ability to exploit effectively this information. Finally, we also provide the performance of a conventional DIIQN agent supplied with the mentor dataset, although they lack the capability to execute identical movements. This additional comparison is intended to underscore the inherent limitations of DIIQN in handling heterogeneous mentor demonstrations, juxtaposed against the efficacy of our HA-DIIQN framework.

We begin by investigating the baseline DQN agent. Initially, its progress is modest, requiring approximately 1700 episodes to grasp fundamental techniques for navigating around walls and boundaries. Afterwards, it gradually ceases

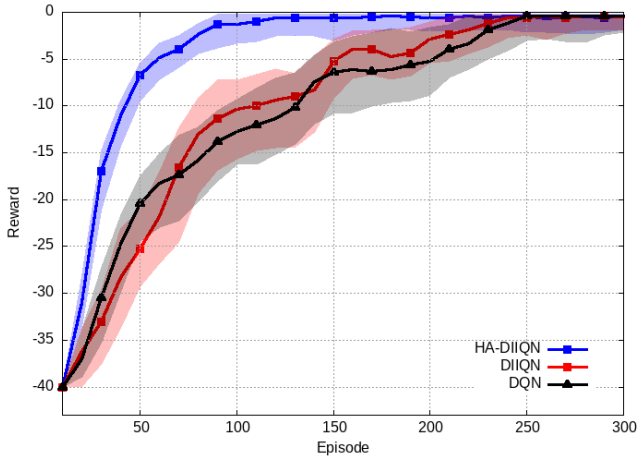


Figure 9: Point Maze: reward per episode comparison, averaged over 30 runs. Shaded area depicts standard deviation.

revisiting states and consistently identifies an optimal path, converging to the target reward of -3.4 by episode 3900.

Following with the learning curve of DIIQN, due to the mentor's utilization of a different action set, we do not anticipate substantial enhancement in training speed. Nevertheless, the agent can capitalize on some of the information shared by the mentor, since parts of the action sets are common—i.e., the $North$ and $South$ directions. However, a considerable portion of the mentor's actions may be irrelevant or potentially misleading. By the inherent nature of the DIIQN's loss function, it can still avert total misguidance and failure to converge, but in the end, the observed acceleration in learning speed with respect to DQN remains marginal, with convergence occurring at approximately episode 3760. It is evident that the ineffective performance of the imitation module can be attributed to its inability to handle heterogeneous action sets appropriately.

Finally, we present the performance of our HA-DIIQN agent. From the really early stages, the agent demonstrates an ability to effectively utilize the heterogeneous dataset by training on bridges aligned with its action set. It promptly navigates past obstacles posed by walls and boundaries and directly seeks to optimize its guided trajectory. Ultimately, the agent achieves convergence at episode 1140, indicating significantly accelerated convergence (compared to the convergence speeds of both the DQN and DIIQN agents).

Point Maze In the Point Maze environment experiments, we employ as the mentor a fully trained agent utilizing DQN and an orthogonal action space (Fig. 7a). On the contrary, all algorithms under evaluation utilize a diagonal action space (Fig. 7b), which shares no actions with the mentor, thereby allowing HA-DIIQN to operate independently.

We present the reward performance of all variants in Fig. 9 over a 300 episode horizon. Here we again test three types of agents: the baseline DQN agent without the incorporation of imitation, the DIIQN agent exposed to heterogeneous mentorship without explicit guidance on how to interpret it, and the HA-DIIQN agent, capable of identifying and navigating around infeasible actions.

In the baseline DQN agent, since the reward function does not provide explicit behavioral information beyond the achievement of reaching the goal, no clear benchmarks can be observed. Instead, the agent demonstrates a smooth and continuous improvement. The agent continuously shortens its path to the target, achieving convergence towards the designated reward of -0.6 per episode at 260 episodes.

In the DIIQN case, the agent cannot benefit from the imitation learning component due to the lack of shared actions with the mentor. Most of the transitions provided by the mentor cannot be explicitly replicated by the observer, rendering them ineffective within the imitation framework of the DIIQN algorithm. Consequently, it is unsurprising that the learning curve of the DIIQN agent exhibits behavior similar to that of the baseline DQN agent, reaching convergence at the 250 episode mark. As previously stated, the DIIQN agent lacks explicit information regarding the feasibility of each mentor transition. The assumption of action feasibility is inherent in the DIIQN algorithm, presupposing that all mentor transitions are viable. As such, the potential for further performance enhancement presents itself.

Indeed, by enabling the heterogeneity-aware HA-DIIQN and its "bridging" mechanism, we can ameliorate the effect that the inherent infeasibility of reproducing the continuous mentor actions has on imitation learning performance; and as such in essence maximize the benefits from imitation RL in this challenging domain. Fig. 9 depicts a learning curve for HA-DIIQN that is considerably steeper than the DIIQN one. HA-DIIQN's learning performance surpasses that of both its competitors, and expedites the optimization of the agent control policy. Specifically, the HA-DIIQN agent attains convergence at episode 110, thereby demonstrating its strong superiority over both DIIQN and standard DQN.

Conclusions and Future Work

In this paper we have put forward a novel framework to enable deep implicit imitation RL in heterogeneous mentor-observer action settings. Our approach is experimentally shown to accelerate learning performance, compared against both a conventional deep RL approach (DQN) and a DIIRL technique (DIIQN) that is not heterogeneity-aware.

Future work endeavours can concentrate on expanding the framework's capabilities and enhance its computational efficiency. For instance, in problem instances necessitating large experience replay sizes while operating with limited computational resources, a viable strategy involves modifying the experience replay search engine by grouping together similar states to facilitate a more efficient search process, potentially incorporating a graph-based approach as proposed in (Eysenbach, Salakhutdinov, and Levine 2019). Moreover, we plan to conduct experiments on other MuJoCo environments, which, although not inherently heterogeneous, can be modified to create such scenarios. In addition, it would be interesting to apply our method in traffic signal control problems—e.g., focusing on controlling distinct sets of non-conflicting signal phases. Finally, we aim to integrate and test our framework's mechanisms in *explicit imitation* (Torabi, Warnell, and Stone 2018, 2019) algorithms.

Acknowledgments

The research described in this paper was carried out within the framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union - NextGenerationEU (Implementation Body: HFRI. Project name: DEEPREBAYES. HFRI Project Number 15430). Moreover, the research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 Research and Innovation programme/ERC Grant Agreement n.[833915], project TrafficFluid.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *ICML*.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*.

Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; and Bharath, A. A. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*.

Bao, R.; Sun, Y.; Gao, Y.; Wang, J.; Yang, Q.; Chen, H.; Mao, Z.-H.; Xie, X.; and Ye, Y. 2023. A Survey on Heterogeneous Transfer Learning. *arXiv preprint arXiv:2310.08459*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. In *arXiv 1606.01540*.

Chrysomallis, I.; Troullinos, D.; Chalkiadakis, G.; Papamichail, I.; and Papageorgiou, M. 2023. Deep Reinforcement Learning with Implicit Imitation for Lane-Free Autonomous Driving. In *ECAI 2023*, 461–468. IOS Press.

Da Silva, F. L.; Warnell, G.; Costa, A. H. R.; and Stone, P. 2020. Agents teaching agents: a survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems*, 34: 1–17.

Day, O.; and Khoshgoftaar, T. M. 2017. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1): 29.

Duan, L.; Xu, D.; and Tsang, I. 2012. Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*.

Duchi, J. C. 2016. Derivations for Linear Algebra and Optimization.

Eysenbach, B.; Salakhutdinov, R. R.; and Levine, S. 2019. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219.

Goldberger, J.; Roweis, S.; Hinton, G.; and Salakhutdinov, R. 2004. Neighbourhood Components Analysis. In *NIPS'04*.

Harel, M.; and Mannor, S. 2010. Learning from multiple outlooks. *arXiv preprint arXiv:1005.0027*.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; Dulac-Arnold, G.; Agapiou, J.; Leibo, J.; and Gruslys, A. 2018. Deep Q-Learning from Demonstrations. In *AAAI*.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation Learning: A Survey of Learning Methods. *ACM Comput. Surv.*, 50(2).

Kang, B.; Jie, Z.; and Feng, J. 2018. Policy Optimization with Demonstrations. In *PMLR'18*.

Lin, L.-J. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.*, 8(3–4).

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2017. Overcoming Exploration in Reinforcement Learning with Demonstrations.

Prettenhofer, P.; and Stein, B. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, 1118–1127.

Price, B.; and Boutilier, C. 1999. Implicit Imitation in Multiagent Reinforcement Learning. In *ICML*, 325–334.

Price, B.; and Boutilier, C. 2001. Imitation and reinforcement learning in agents with heterogeneous actions. In *Advances in Artificial Intelligence: 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI 2001 Ottawa, Canada, June 7–9, 2001 Proceedings 14*, 111–120. Springer.

Price, B.; and Boutilier, C. 2003. Accelerating Reinforcement Learning through Implicit Imitation. *Journal of Artificial Intelligence Research*.

Silver et al., D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral Cloning from Observation. arXiv:1805.01954.

Torabi, F.; Warnell, G.; and Stone, P. 2019. Adversarial Imitation Learning from State-Only Demonstrations. In *AAMAS '19*.

Tsai, Y.-H. H.; Yeh, Y.-R.; and Wang, Y.-C. F. 2016. Learning cross-domain landmarks for heterogeneous domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5081–5090.

Vecerik, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2017. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards.

Wang, C.; and Mahadevan, S. 2011. Heterogeneous domain adaptation using manifold alignment. In *IJCAI proceedings-international joint conference on artificial intelligence*, volume 22, 1541.

Wang, C.; Warnell, G.; and Stone, P. 2023. D-Shape: Demonstration-Shaped Reinforcement Learning via Goal-Conditioning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, 1267–1275. ACM.

Yan, Y.; Wu, Q.; Tan, M.; and Min, H. 2016. Online heterogeneous transfer learning by weighted offline and online classifiers. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, 467–474. Springer.

Yeh, Y.-R.; Huang, C.-H.; and Wang, Y.-C. F. 2014. Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *IEEE Transactions on Image Processing*, 23(5): 2009–2018.

Zhang, R.; Torabi, F.; Warnell, G.; and Stone, P. 2021. Recent advances in leveraging human guidance for sequential decision-making tasks. *Autonomous Agents and Multi-Agent Systems*, 35(2): 31.

Zhao, P.; and Hoi, S. 2010. Otl: A framework of online transfer learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010): Haifa, Israel, 21-24 June*, 219–1.

Zhou, J.; Pan, S.; Tsang, I.; and Yan, Y. 2014. Hybrid heterogeneous transfer learning through deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.