ModelDiff: Symbolic Dynamic Programming for Model-Aware Policy Transfer in Deep Q-Learning

Xiaotian Liu[1] Jihwan Jeong[1], Ayal Taitler[1,3], Michael Gimelfarb[1], Scott Sanner[1,2]
[1]University of Toronto, Toronto, ON, Canada
[2]Vector Institute for AI, Toronto, ON, Canada
[3]Ben-Gurion University of the Negev, Be'er Sheva, Israel
xiaotian.liu@mail.utoronto.ca, jhjeong@mie.utoronto.ca, ataitler@bgu.ac.il, mike.gimelfarb@mail.utoronto.ca, ssanner@mie.utoronto.ca

Abstract

Despite significant recent advances in the field of Deep Reinforcement Learning (DRL), such methods typically incur high cost of training to learn effective policies, thus posing cost and safety challenges in many practical applications. To improve the learning efficiency of (D)RL methods, transfer learning (TL) has emerged as a promising approach to leverage prior experience on a source domain to speed learning on a new, but related, target domain. In this paper, we take a novel model-informed approach to TL in DRL by assuming that we have knowledge of both the source and target domain models (which would be the case in the prevalent setting of DRL with simulators). While directly solving either the source or target MDP via solution methods like value iteration is computationally prohibitive, we exploit the fact that if the target and source MDPs differ only due to a small structural change in their rewards, we can apply structured value iteration methods in a procedure we term ModelDiff to solve the much smaller target−source "Diff" MDP for a reasonable horizon. This ModelDiff approach can then be integrated into extensions of standard DRL algorithms like ModelDiff (MD) DQN, where it provides enhanced provable lower bound guidance to DQN that often speeds convergence for the positive transfer case while critically avoiding decelerated learning in the negative transfer case. Experiments show that MD-DQN matches or outperforms existing TL methods and baselines in both positive and negative transfer settings.

Introduction

Deep Reinforcement Learning (DRL) algorithms typically require extensive agent experience to develop effective policies (Mnih et al. 2015; Zhu, Lin, and Zhou 2020), thus posing cost and safety challenges in many practical applications (Gu et al. 2022). To tackle the efficiency problem inherent in (D)RL, transfer learning (TL) has (Taylor and Stone 2009; Zhu, Lin, and Zhou 2020) emerged as a promising approach aiming to leverage prior knowledge for learning new tasks. TL capitalizes on the idea that an agent equipped with prior knowledge from a previously learned task can rapidly adapt and learn on a new task.

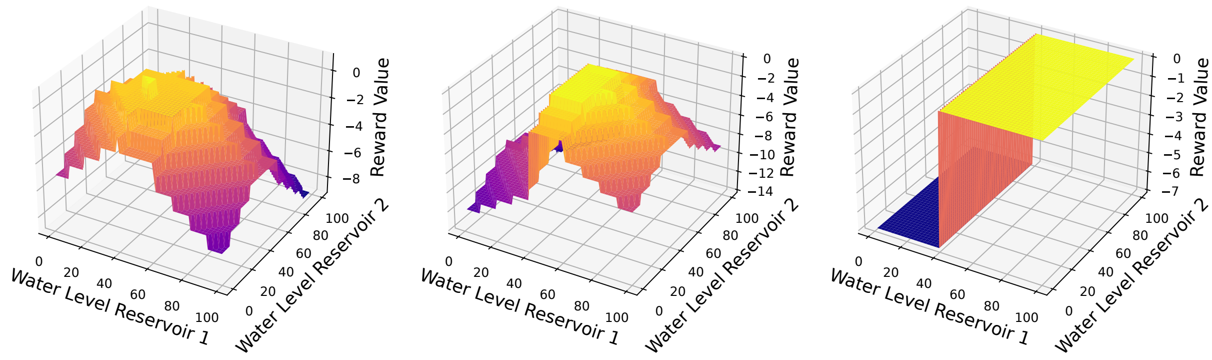
Ideally, transfer learning (TL) techniques should transfer only knowledge that is beneficial (positive transfer) while avoiding knowledge that could hinder learning (negative transfer). However, the existing TL literature often treats positive and negative transfer as separate problems (Zhu, Lin, and Zhou 2020; Zhang et al. 2023). Most TL techniques focus on positive transfer by learning a diverse set of policies (Fernández and Veloso 2006) or features (Barreto et al. 2018) that can generalize to new but similar tasks. The problem of negative transfer is studied separately, often through techniques such as regularization (Jung et al. 2020) and data selection (Wang and Carbonell 2018).

Developing methods that are robust to negative transfer is crucial for real-world applications, where safety is a major concern. Moreover, access to multiple source policies or extensive training data is often impractical (Vettoruzzo et al. 2023; Tian et al. 2020). A frequently overlooked aspect is that exact or analytic models of the environment are available for many RL applications, and existing works fail to fully leverage this knowledge (Dulac-Arnold et al. 2021). Thus there is a significant opportunity for more targeted transfer learning methods using model knowledge that have the ability to avoid negative transfer.

In this work, we focus on scenarios where the source and target domains are known and dynamically similar but their rewards differ in structurally simple ways. As an illustration, Figure 1 shows the reward functions of R_s, R_t, and R_d in a two-reservoir RESERVOIR MANAGEMENT problem. Here, the "diff" reward R_d is an extra penalty for being below a desired water-level, which is far simpler than the individual rewards. This simplification is a also common occurrence in practical applications, such as in robotics, where tasks often include a complex smoothness penalty alongside a straightforward goal-conditioned reward. For different tasks with varied goals, the smoothness penalty would remain consistent, resulting in a diff reward that can be expressed in a much simpler analytic form. This scenario raises a pivotal research question: *how can we effectively leverage the reward difference for efficient transfer learning?*

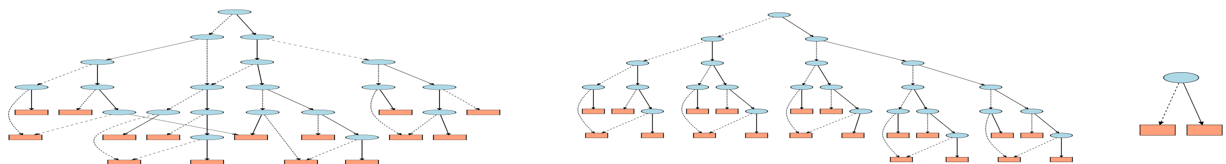
To address this challenge, we introduce **ModelDiff**, a novel TL framework designed to utilize analytically available models for efficient policy transfer from a single source to a target domain. We use a compact symbolic representation of the underlying MDPs using Extended Algebraic Decision Diagrams (XADDs) (Sanner, Delgado, and de Barros 2011). Central to ModelDiff is the exploitation of the simpler *difference MDP*, which can be calculated directly via

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) Source reward: R_s (b) Target reward: R_t (c) "Diff" reward: R_d

Figure 1: Source and target reward in RESERVOIR MANAGEMENT where reservoirs are penalized based on absolute distance to the desired water-level. The simpler "Diff reward" owing to a minor structural change of extra penalty for going below the desired water-level.



(a) Source R_s XADD (b) Target R_t XADD (c) 'Diff" R_d XADD

Figure 2: Symbolic Representation (XADD) of Source, Target, and Diff Rewards for the RESERVOIR MANAGEMENT problem. Enlarged versions of the XADDs depicted in (a) and (b) are provided in Appendix D for detailed inspection.

symbolic subtraction. An example of the difference reward XADD can be found in Figure 2. The symbolic representation then allows for the exact evaluation of the source policy under this simpler ModelDiff difference MDP via Symbolic Dynamic Programming (SDP) (Hoey et al. 1999), leading to the derivation of a compact symbolic lower bound Q-value for the target domain. We develop a modified Bellman operator to evaluate this symbolic lower bound Q-value and clip each Q-learning update at its lower bound to expedite knowledge transfer from the source to the target domain.

The efficacy of ModelDiff is showcased through its integration with a standard model-free Q-learning RL algorithm, DQN (Mnih et al. 2015), across three distinct single-source transfer learning scenarios. A key advantage of ModelDiff lies in the exactly evaluated value function of the policy, which serves as a lower bound for the DQN value targets. This lower bound acts as a robust learning signal, enhancing efficient exploration in cases of positive transfer. Importantly, it also ensures that the learning process is not adversely affected in scenarios of negative transfer since the lower bound is often exceeded and thus ignored.

We summarize three of our main contributions as follows:

1. We derive a novel Bellman-style operator on Q-value functions that provides a lower bound estimate on the optimal Q-value Q^*, and we prove its convergence to Q^*. We also establish a value iteration-style recurrence for the difference model that lower bounds Q^*.
2. Based on the above derivation, we introduce ModelDiff to provide lower bound guidance for DQN agents (i.e., MD-DQN). ModelDiff-informed MD-DQN enhances learning efficiency in three experimental TL tasks compared to related work and baselines.
3. Lastly, we show that ModelDiff-informed MD-DQN effectively avoids negative transfer, where the source policy might adversely affect learning in the target domain.

Preliminaries

Reinforcement Learning

In the reinforcement learning (RL) setting, an agent acts in an environment to optimize its performance based on a given reward function. We model the environment as a Markov Decision Process (MDP), defined by a tuple $\langle S, A, T, R, \gamma \rangle$. Here, S is the set of states, A is the set of valid actions, T is a Markovian transition function that describes the distribution $p(s'|s, a)$ of the next state s' given the current state s and action a, and r is the immediate reward function. The discount factor $\gamma \in (0, 1)$ ensures that rewards received far into the future receive less weight than immediate rewards.

The objective of an RL agent is to learn a policy $\pi : S \rightarrow A$ that maximizes the expected (discounted) sum of future rewards $Q^\pi(s, a) = \mathbb{E}_{s_t} [\sum_t \gamma^t R(s_t, \pi(s_t)) \,|\, s_0 = s,\, a_0 = a]$. Value-based

RL aims to learn the optimal Q-values Q^* that are the fixed point $Q^* = \mathcal{B}(Q^*)$ of the Bellman operator

$$\mathcal{B}(Q) = \mathbb{E}_{s'}\left[R(s,a) + \gamma \max_{a'} Q(s',a')\right], \qquad (1)$$

which is a γ-contraction over the space of Q-value functions.

Q-learning (Sutton and Barto 2018) is a popular value-based RL algorithm that learns the Q value function via the so-called temporal difference (TD) update

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)].$$

Deep Q-Network (DQN) is a neural net-based version of Q-learning that learns a neural network function approximator of the Q function (Mnih et al. 2015; van Hasselt, Guez, and Silver 2015).

Symbolic Dynamic Programming (SDP)

Factored MDPs (Boutilier, Dean, and Hanks 1999) share the same components $\langle S, A, T, R, \gamma\rangle$ as MDPs. However, they assume that the state and dynamics are factorized in terms of a vector of discrete and continuous state variables. Symbolic Dynamic Programming (SDP) (Hoey et al. 1999) is a Value Iteration algorithm designed to compute Bellman backups in (1) so as to provide *compact, structured* representations of the Q-values as Extended Algebraic Decision Diagrams (XADDs) in both the mixed discrete and continuous state and action settings (Sanner, Delgado, and de Barros 2011; Zamani, Sanner, and Fang 2012). Examples of symbolically represented reward functions represented as XADDs are shown in Figure 2. We refer the reader to (Sanner, Delgado, and de Barros 2011; Zamani, Sanner, and Fang 2012) for more details on SDP operations on XADDs.

Related Work

Knowledge transfer in RL can take the form of polices, value functions, state transitions, rewards, or other sources (Zhu, Lin, and Zhou 2020). One common method is Probabilistic Policy Reuse (PPR) (Fernández and Veloso 2006), where exploration is biased by assigning a probability to act according to a set of policies learned from similar tasks. Potential based Reward Shaping (PBRS) (Ng, Harada, and Russell 1999) is another common technique that biases the reward function on a target domain (Brys et al. 2015; Vecerík et al. 2017). Our work also shares some characteristics with Successor Features-based transfer (GPI&SF) (Barreto et al. 2017, 2018), which assume all task rewards are linear combinations of shared reward components. However, while the latter requires a restricted reward space (i.e., linear combinations of task rewards) and the model-free setting, our work assumes the reward function is known and leverages the structural reward difference in (arbitrary) source and target rewards for transfer. (Song et al. 2016) proposed a method to select suitable policies and value functions for transfer via a distance measure among MDPs. (Brys et al. 2015) used reward shaping techniques to transfer policies via inter-task state space matching, while (Gimelfarb, Sanner, and Lee 2021) used transition function similarity. Meanwhile, (Lazaric, Restelli, and Bonarini 2008) and (Laroche and Barlier 2017) reused previous source task interactions by utilizing task and reward similarities. Work in this area often focused on using models to select the best policies for transfer. However, we focus on evaluating performance of any given policy in the target domain and the difference between models. Another key aspect often ignored by previous work is the high computational overhead required to reason over large models. Our work aims to reduce computational overhead required by analyzing model differences.

Methodology

Problem Setup

We assume two MDPs $M_s = \langle S, A, T, R_s, \gamma\rangle$ and $M_t = \langle S, A, T, R_t, \gamma\rangle$ respectively designated as the source and target tasks that share identical S, A, and T, but differ in their reward structure. We also assume that we have a (good) policy π_s and its corresponding Q-value $Q_s^{\pi_s,h}(s,a)$, evaluated for horizon h on M_s. Our goal is to leverage π_s, $Q_s^{\pi_s,h}(s,a)$, and M_s to accelerate DQN learning on M_t when M_s and M_t share similar optimal policies, but to avoid negative transfer and decelerated DQN learning otherwise.

To achieve this goal, we leverage the key insight that a *lower bound* on $Q_t^{\pi_t^*,h}(s,a)$ can *accelerate* DQN learning for M_t in the case of *positive transfer* where the lower bound is informative, while *preventing negative transfer* where the lower bound is exceeded and therefore correctly ignored. What remains is to efficiently derive this lower bound and show how to use it in a modified DQN for transfer learning.

A Lower Bound on the Optimal Q-Value

A natural way to lower bound $Q_t^{\pi_t^*,h}(s,a)$ is by noting that $Q_t^{\pi_s,h}(s,a) \le Q_t^{\pi_t^*,h}(s,a)$ since π_t^* is optimal for M_t. While we cannot compute $Q_t^{\pi_s,h}(s,a)$ directly via SDP due to the reward complexity of M_t (cf. Figures 1,2), we can instead compute it efficiently by leveraging the model difference (ModelDiff) between M_t and M_s and the known π_s and $Q_s^{\pi_s,h}(s,a)$. But first we introduce two helpful definitions:

Definition 1 (Difference Reward)**.** *We define the difference in reward functions between M_s and M_t for state s and action a as **Difference Reward**:*

$$R_d(s,a) = R_t(s,a) - R_s(s,a). \qquad (2)$$

Definition 2 (Difference Q-value)**.** *Given a given source policy π_s and a fixed horizon h, we can define **Difference Q-value** as:*

$$Q_d^{\pi_s,h}(s,a) = Q_t^{\pi_s,h}(s,a) - Q_s^{\pi_s,h}(s,a) \qquad (3)$$

Where $Q_s^{\pi,h}(s,a)$ and $Q_t^{\pi,h}(s,a)$ are value functions evaluated on M_s and M_t respectively under π for horizon h.

By simply rearranging these equations in Definition 2, we can directly obtain our intended lower bound $Q_t^{\pi_s,h}(s,a)$:

$$Q_t^{\pi_s,h}(s,a) = Q_s^{\pi_s,h}(s,a) + Q_d^{\pi_s,h}(s,a). \qquad (4)$$

Algorithm 1: ModelDiff DQN (MD-DQN)

Initialize replay memory D with capacity N
Initialize Q-network with weights θ
Initialize target network with weights $\theta^- = \theta$
for episode $= 1, \ldots, M$ **do**
  Initialize state s
  for $t = 1, \ldots, H$ **do**
    With probability ϵ, select a random action a
    Otherwise select $a = \arg\max_{a'} Q(s, a'; \theta)$
    Execute action a, observe reward r and next state s'
    Store transition (s, a, r, s') in D
    Sample a batch $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{J}$ from D
    Compute target for all $j \in \{1, \ldots, J\}$:

$$y_j \leftarrow \max\left(r_j + \gamma \max_{a'} Q(s'_j, a'; \theta^-), Q_t^{\pi_s}(s_j, a_j)\right)$$

    Update $\theta \leftarrow \theta - \frac{\eta}{J} \nabla_\theta \sum_j (y_j - Q(s_j, a_j; \theta))^2$
    Update target network $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$
    Set $s \leftarrow s'$
  end for
end for

Again, we know $Q_s^{\pi_s,h}(s,a)$ so we need only compute $Q_d^{\pi_s,h}(s,a)$, which can be recursively evaluated as follows:

$$\begin{aligned}
&Q_d^{\pi_s,h}(s,a) = Q_t^{\pi_s,h}(s,a) - Q_s^{\pi_s,h}(s,a) \\
&= R_t(s,a) + \gamma \mathbb{E}_{s'}\left[P(s'|s,a) Q_t^{\pi_s,h-1}(s', \pi_s(s'))\right] \\
&\quad - R_s(s,a) - \gamma \mathbb{E}_{s'}\left[P(s'|s,a) Q_s^{\pi_s,h-1}(s', \pi_s(s'))\right] \\
&= \underbrace{R_t(s,a) - R_s(s,a)}_{R_d(s,a)} \\
&\quad + \gamma \mathbb{E}_{s'}\left[P(s'|s,a)\underbrace{[Q_t^{\pi_s,h-1}(s',\pi_s(s')) - Q_s^{\pi_s,h-1}(s',\pi_s(s'))]}_{Q_d^{\pi_s,h-1}(s',\pi_s(s'))}\right]
\end{aligned}$$

In short, $Q_d^{\pi_s,h}(s,a)$ can be computed via a Bellman recursion on the ModelDiff, i.e., $R_d(s,a)$. However, recall that R_d defines an MDP with a much simpler reward function than R_s and R_t (cf. Figures 1,2), thus enabling *efficient* SDP computation of this ModelDiff Bellman backup that can directly exploit the compactness of R_d. This yields efficient computation of the lower bound $Q_t^{\pi_s,h}(s,a)$ on M_t via (4)!

Incorporating the Lower Bound into RL

A remaining key question is how to incorporate the ModelDiff lower bound $Q_t^{\pi_s}(s,a)$ into the Q-learning update to accelerate target task training. To answer this question, we first define the following modified Bellman operator $\mathcal{L}_f$:

$$\begin{aligned}
&\mathcal{L}_f(Q)(s,a) \\
&= \max\left(\mathcal{B}(Q)(s,a), f(s,a)\right) \\
&= \max\left(\mathbb{E}_{s'}\left[R(s,a) + \gamma \max_{a'} Q(s',a')\right], f(s,a)\right), \quad (5)
\end{aligned}$$

where $f(s,a)$ is a lower bound on Q^*, i.e. $f(s,a) \leq Q^*(s,a), \forall s,a$. Intuitively, if f is close to Q^*, then updating the Q-value using $\mathcal{L}_f$ will drive the Q-value estimates closer to Q^* than when using $\mathcal{B}$. This means we can interpret f as a source of positive knowledge transfer. Stated more formally:

Theorem 1. *Starting with Q_0, and f such that $f(s,a) \leq Q^*(s,a), \forall s,a$, the sequence Q_k produced by $Q_{k+1} = \mathcal{L}_f(Q_k)$ converges to the optimal value function Q^* in the usual sup-norm $\|\cdot\|_\infty$, i.e. the fixed point $Q^* = \mathcal{B}(Q^*)$. Further, value iteration using $\mathcal{L}_f$ converges to Q^* at least as fast as $\mathcal{B}$.*

Thus, (5) can be easily incorporated into a Q-learning update by using our established lower bound $Q_t^{\pi_s}(s,a)$ in place of $f(s,a)$! More concretely, the Q-learning update rule for each state-action pair becomes:

$$\begin{aligned}
&Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [y_{target} - Q(s,a)] \\
&y_{target} = \max\left(r(s,a) + \gamma \max_{a'} Q(s',a'), Q_t^{\pi_s}(s,a)\right).
\end{aligned}$$

Intuitively, the lower bound forces the agent to act according to $Q_t^{\pi_s}(s,a)$ in the initial steps of training, thus providing a "model-informed" warm-start. In the later stages of training, when it is anticipated that the target agent's Q-function is already a good approximation of $Q_t^{\pi_t^*}(s,a)$, the target value above approaches the usual Q-learning target.

To make the algorithm scale well in large state or action spaces, we parameterize the Q-function as a Deep Q-Network (DQN) (Mnih et al. 2015). Let $Q(s,a;\theta) \approx Q(s,a)$ be a neural network with weights θ. The modified DQN loss for a transition (s,a,s'), derived from $\mathcal{L}_{Q_t^{\pi_s}}$, is:

$$\begin{aligned}
&L(\theta) = \frac{1}{2}\left(y_{target} - Q(s,a;\theta)\right)^2, \\
&y_{target} = \max\left(r(s,a) + \gamma \max_{a'} Q(s',a';\theta^-), Q_t^{\pi_s}(s,a)\right)
\end{aligned}$$

where θ^- indicates that the gradient ∇_θ does not propagate through $Q(s',a';\theta^-)$. The MD-DQN implementation using the ModelDiff lower bound can be found in Algorithm 1.

Proof of Theorem 1. First, observe that for any k, we have

$$\begin{aligned}
&|\max(\mathcal{B}(Q_k)(s,a), f(s,a)) - Q^*(s,a)| \\
&\leq |\mathcal{B}(Q_k)(s,a) - Q^*(s,a)|.
\end{aligned}$$

To prove this, observe that if $f(s,a) \geq \mathcal{B}(Q_k)(s,a)$, then

$$\begin{aligned}
0 &\leq Q^*(s,a) - \max(\mathcal{B}(Q_k)(s,a), f(s,a)) \\
&\leq Q^*(s,a) - \mathcal{B}(Q_k)(s,a).
\end{aligned}$$

On the other hand, if $f(s,a) < \mathcal{B}(Q_k)(s,a)$, then

$$\begin{aligned}
&Q^*(s,a) - \max(\mathcal{B}(Q_k)(s,a), f(s,a)) \\
&= Q^*(s,a) - \mathcal{B}(Q_k)(s,a).
\end{aligned}$$

Using this result, we continue as follows:

$$\begin{aligned}
&|Q_{k+1}(s,a) - Q^*(s,a)| \\
&= |\mathcal{L}_f(Q_k)(s,a) - Q^*(s,a)| \\
&= |\max\left(\mathcal{B}(Q_k)(s,a), f(s,a)\right) - Q^*(s,a)| \\
&\leq |\mathcal{B}(Q_k)(s,a) - Q^*(s,a)| \\
&= |\mathcal{B}(Q_k)(s,a) - \mathcal{B}(Q^*)(s,a)| \\
&\leq \gamma \|Q_k - Q^*\|_\infty,
\end{aligned}$$

where in the last step we have used the fact that the Bellman operator $\mathcal{B}$ is a γ-contraction. Thus, taking sup of both sides over s, a, we conclude that

$$\|Q_{k+1} - Q^*\|_\infty \le \gamma \|Q_k - Q^*\|_\infty,$$

and thus that $Q_k \to Q^*$ as $k \to \infty$, as claimed.

Finally, the last argument in the proof, namely the inequality $|Q_{k+1}(s,a) - Q^*(s,a)| \le |\mathcal{B}(Q_k)(s,a) - Q^*(s,a)|$, shows (by induction) that value iteration using $\mathcal{L}_f$ converges to Q^* at least as fast as $\mathcal{B}$. □

Symbolic Policy Extraction

For scenarios where the source policy is manually specified, we can directly convert such a policy into a symbolic representation like an XADD. However, in cases where the source policy is implicitly represented in a function approximator like a DQN, we can obtain a sufficiently good approximation of the source policy by sampling state-action pairs from the neural network. We then recover a decision tree (DT) via generated samples, and convert the DT to an XADD format. The details of this procedure are discussed in the Appendix A. We found that we can extract a policy that sufficiently mimics the exact behavior in the reachable states of the agent given sufficient samples.

Empirical Evaluation

We evaluated ModelDiff-informed MD-DQN (from here out, shortened to just MD-DQN) derived lower bound on three different domains in a transfer learning setting with source and target tasks in each. All the domains and tasks have been implemented and trained in *pyRDDLGym* (Taitler et al. 2022). The experiments are designed to answer two research questions:

1. **RQ 1:** How much can MD-DQN accelerate transfer vs. SOTA transfer methods?
2. **RQ 2:** Is MD-DQN more robust to preventing negative transfer than SOTA transfer methods?

Experimental Setup

We evaluate MD-DQN on three benchmark domains: POWERGEN, PICK-AND-PLACE, and RESERVOIR. For each domain, we know the analytic forms of M_s and M_t, and a source policy to be transferred is given. Specifically, we use a pre-trained neural policy derived by a DQN agent on M_s and leverage it in the ModelDiff framework to derive the lower bound for M_t.

For each domain, we define two target tasks that share the same states, actions, and transition function as in M_s but have different reward functions. The first target task is designed to test *positive transfer*; therefore, the difference in the reward functions is small, and the source policy can achieve high rewards in all but a small subset of states in the target task. The second target task is used to test *negative transfer*. In this scenario, the source policy behaves adversarially in the target task. We aim to assess if MD-DQN can effectively disregard the adversarial behavior of this suboptimal policy. The detailed transition and rewards of the domains can be bound in Appendix B.

All the experiments are compared to baseline methods that are appropriate for our experimental setting:

- **DQN** – the vanilla model-free DQN that learns from scratch on the target domain.
- **Probabilistic Policy Reuse (PPR)** (Fernández and Veloso 2006) – a DQN-based method that assigns a probability of reusing the source policy for guidance based on an assessment of its relevance to the target policy.
- **Warm Start (WS)** – DQN with the value function being directly initialized with the source value function.

In the positive transfer experiments, all methods are expected to learn more efficiently than vanilla DQN. In the negative transfer experiments, success is instead measured by the method's ability to be at least as good as DQN that intrinsically ignores transfer (i.e., optimal in this setting).

Benchmark Domains

RESERVOIR Drawing from the water reservoir management problem (Yeh 1985), we set up a 2-reservoir system with one upstream reservoir t_1 and one downstream reservoir t_2. The water released from t_1 flows to t_2, and the water released from t_2 is directly released to the ocean and leaves the system. The objective is to maintain the water levels of both reservoirs within a maximum and minimum threshold. We give a negative reward if the water level goes outside the given thresholds. At each time step, there is a chance for rainfall, modeled by a stochastic Bernoulli random variable (RV). When it rains, a fixed amount of water is added to the reservoirs. The RV of the first reservoir is independent of the variable of the other reservoir. We have a Boolean decision for each reservoir, dictating whether to release water. For the positive transfer case, we reduce the penalty for being below the minimal threshold by half for the upstream reservoir t_1, which should only slightly alter optimal actions in the new task. For the negative transfer case, we move both the maximum and the minimum threshold. For most states in the target task, optimal decisions differ from the source task.

PICK-AND-PLACE In this domain, an agent needs to discover an object of interest, pick it up, and place it in another location. The problem is inspired by a 2D navigation problem specified in (Taitler et al. 2022). We added a discrete state variable *has-object* in this domain to indicate if the agent holds the object. The agent can *pick-up* an object if it is in the vicinity of the agent. A reward of 1 is given if the agent is holding the object in one of the two goal regions, and 0 otherwise. The agent can move either in the x or y direction with a fixed step size. For positive transfer, we increase the reward of one of the goal locations. For negative transfer, we reverse the reward of the two goal locations.

POWERGEN We use a modified version of the Power Unit Commitment problem (Padhy 2004) as the third benchmark domain. Here, it models 3 power producers that act cooperatively to meet daily demand. The demand for each timestep is a Bernoulli RV. The agent receives a reward for each demand successfully met. If the demand is not met, then a large penalty occurs to deter overproduction. For the positive transfer case, we increase the production cost of a single

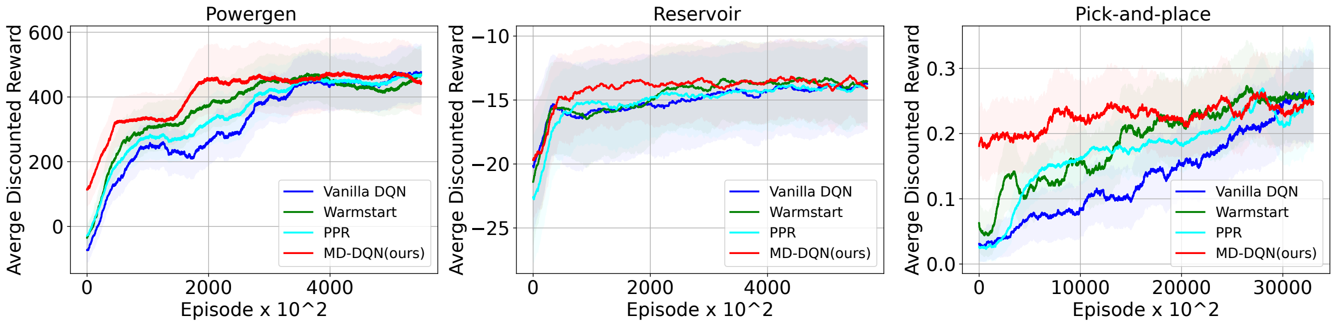


Figure 3: Average cumulative discounted rewards in Positive Transfer tasks

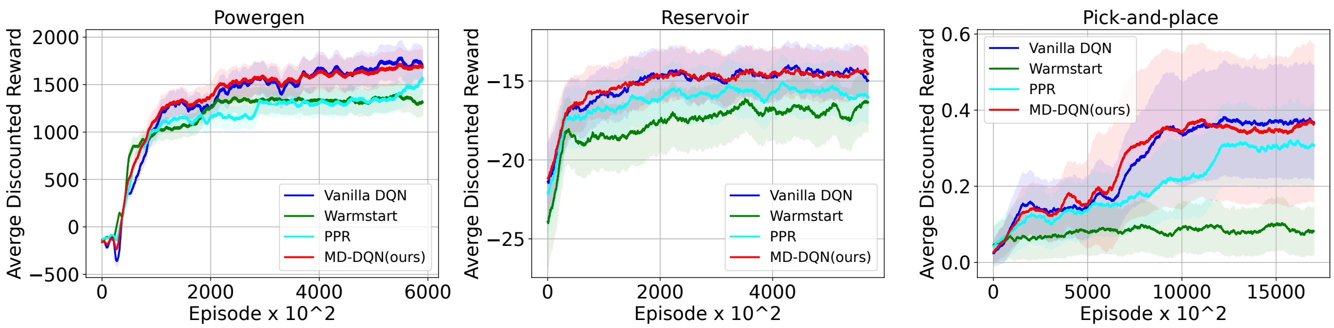


Figure 4: Average cumulative discounted rewards in Negative Transfer tasks

power producer. For the negative transfer case, we increase the production cost of all power producers.

Training Details

We conducted a hyper-parameter search for a vanilla DQN agent across each of the three domains, utilizing the source tasks. The selected hyper-parameters are given in Appndix C. These hyper-parameter values are consistently used by all methods during the learning phase on the target tasks including MD-DQN . Each task is set with a horizon of 20, and the discount factor γ is fixed at 0.9. To evaluate and record the agent's performance, we freeze network weights after every 10 training episodes.

Empirical Results

For all the experiments conducted in this work, 10 runs were executed, and both the average and standard deviation were calculated. In the provided plots, the bold colored lines represent the average, while the grayed-out envelopes indicate the standard deviation across the 10 runs.

Positive Transfer Results The results for the positive transfer experiments for all three domains are given in Figure 3. These scenarios are designed such that a large portion of the policy of the source task can be used on the target task. Consequently, all three transfer methods outperform the vanilla DQN on all three domains. We observe that MD-DQN outperforms all other transfer baselines, i.e., the reward is maximized faster in MD-DQN , while ultimately all the methods converge to the same maximum rewards, illustrating the sample complexity of each method. The benefit of using the explicit model directly, is showcased especially in the *pick-and-place* domain where the initial reward at the beginning of learning is higher than all the others by a significant gap. The overall results indicate that leveraging model knowledge can improve TL performance without the need for mutiple source policies.

Negative Transfer Results Our results demonstrate a clear advantage of MD-DQN over other compared methods in cases of negative transfer. When the target task and the source task differ in such a way that a policy suited for the source task hinders the target task, MD-DQN outperforms other transfer methods. Both PPR and Warmstart significantly impede learning in the target domain compared to the vanilla DQN agent. The negative transfer results are illustrated in Figure 4. Additionally, we observe that using an adversarial policy may not only delay performance but can also prevent the algorithm from discovering a viable policy altogether. This is evident in the *pick-and-place* experiment, where Warmstart becomes clearly stuck. This finding suggests that warm-starting with an adversarially initialized policy value can be even more detrimental than starting with a randomly initialized value (i.e., network weights) and learning from scratch. Across all three domains, MD-DQN performs on par with DQN, indicating that while it may not surpass DQN, it effectively mitigates the effects of negative transfer. In contrast, both PPR and Warmstart consistently suffer from negative transfer across all three domains.

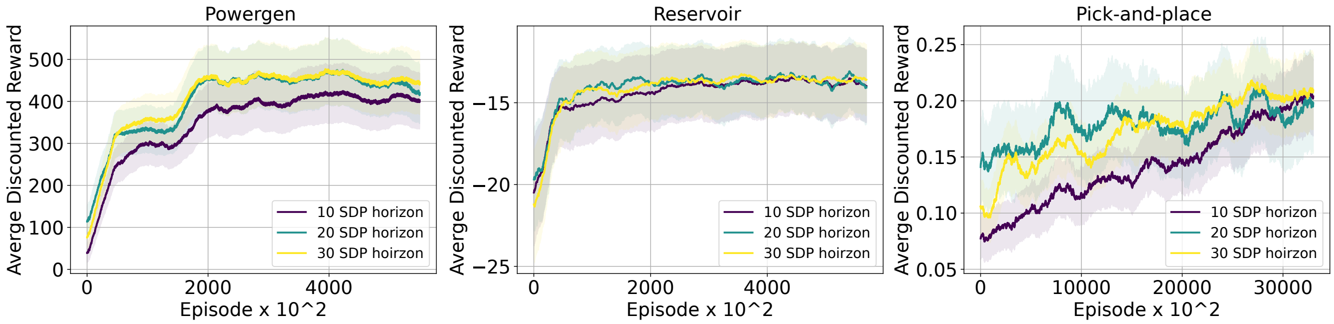


Figure 5: Ablation of the number of SDP horizon's impact on training performance for horizon 10, 20 and 30

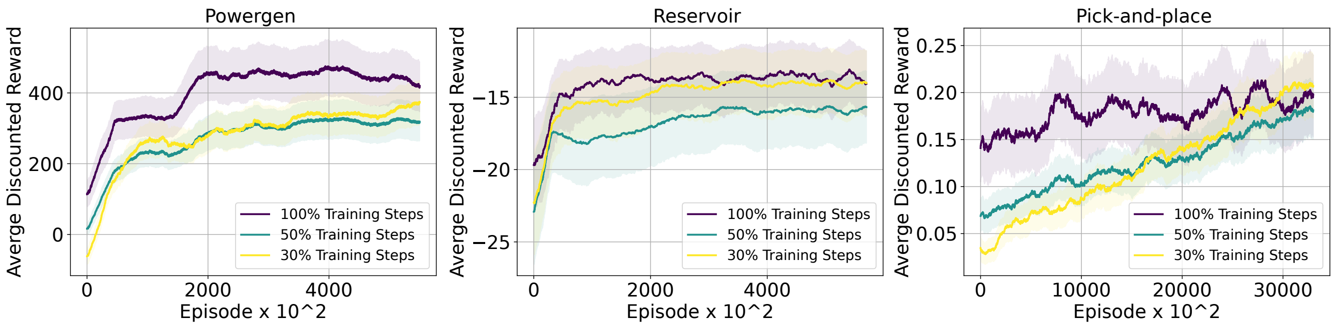


Figure 6: Impact of the source policy quality π_s on performance. π_s are trained to (100%, 50%, 30%) to convergence

Ablations

As ModelDiff-informed MD-DQN is comprised of several components and parameters, such as the source trained policy and Q-network for M_s and the ModelDiff SDP evaluation horizon, we conduct two experiments designed at assessing the sensitivity of these components. Both experiments are conducted on the positive transfer scenario to isolate the desired effects without the influence of negative transfer.

SDP Horizon The lower bound $Q_t^{\pi_s}$, derived from the ModelDiff is computed via SDP as described in our methodology section. We examine how MD-DQN performs vs. $H \in \{10, 20, 30\}$ as shown in Figure 5. We note that increasing the horizon to 30 does not yield significantly better results. Also, while the performance of MD-DQN under 10 SDP steps is inferior to the other runs, we note that the difference is not significant, suggesting that the number of SDP steps required for the lower bound calculation does not need to be large. However, the value of H that optimally trades off performance and evaluation quality can vary between between domains.

Source Policy Quality Key components required for transfer include the reward diff R_d, a source policy π_s for M_s and its corresponding Q-value function. In principle, any π_s can be used since it would provide a lower bound value for M^t (tight, if serendipitously optimal). Nonetheless, policy quality is expected to impact the performance of MD-DQN. Hence, we train DQN in three regimes: (1) to optimality, (2) training for half the time required to achieve optimality, and (3) training for 30% of the time required to achieve optimality. The results for this study are given in Figure 6. MD-DQN is able to effectively transfer from all three policies and Q-functions, although we note that the best performance is obtained by using the optimal source task policy, as expected in this setting of positive transfer.

Conclusion

In this paper, we asked how we could leverage model knowledge of a source and target MDP with matching dynamics but differing reward functions to facilitate transfer learning in a deep reinforcement learning (DRL) setting. We exploited the structural differences between the reward functions to symbolically derive a lower bound on the target MDP. Our experimental results demonstrate that our ModelDiff-informed MD-DQN learns faster than other transfer learning techniques in the positive transfer setting and is critically able to ignore negative transfer in contrast to competing methods. Ablation studies further show that increasing the ModelDiff horizon has positive, but diminishing returns, while source MDP policy quality has a significant impact on transfer efficiency. Overall, this work opens a novel and effective direction for model-informed transfer to help address longstanding sample complexity issues in DRL and inspire novel model-informed transfer extensions beyond MD-DQN.

References

Barreto, A.; Borsa, D.; Quan, J.; Schaul, T.; Silver, D.; Hessel, M.; Mankowitz, D. J.; Zídek, A.; and Munos, R. 2018. Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, 510–519. PMLR.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017. Successor features for transfer in reinforcement learning. *NeurIPS*, 30.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11: 1–94.

Brys, T.; Harutyunyan, A.; Taylor, M. E.; and Nowé, A. 2015. Policy Transfer using Reward Shaping. In *AAMAS*, 181–188. ACM.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Mach. Learn.*, 110(9): 2419–2468.

Fernández, F.; and Veloso, M. M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*, 720–727. ACM.

Gimelfarb, M.; Sanner, S.; and Lee, C.-G. 2021. Contextual policy transfer in reinforcement learning domains via deep mixtures-of-experts. In *UAI*, 1787–1797. PMLR.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; and Knoll, A. 2022. A Review of Safe Reinforcement Learning: Methods, Theory and Applications. *arXiv preprint arXiv:2205.10330.*

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic Planning using Decision Diagrams. In *UAI*, 279–288. Stockholm.

Jung, S.; Ahn, H.; Cha, S.; and Moon, T. 2020. Continual Learning with Node-Importance based Adaptive Group Sparse Regularization. In *NeurIPS*.

Laroche, R.; and Barlier, M. 2017. Transfer Reinforcement Learning with Shared Dynamics. In *AAAI*, 2147–2153. AAAI Press.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Transfer of samples in batch reinforcement learning. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, 544–551. ACM.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; 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: 529–533.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *ICML*, 278–287. Morgan Kaufmann.

Padhy, N. P. 2004. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2): 1196–1205.

Sanner, S.; Delgado, K. V.; and de Barros, L. N. 2011. Symbolic Dynamic Programming for Discrete and Continuous State MDPs. In *UAI*, 643–652. AUAI Press.

Song, J.; Gao, Y.; Wang, H.; and An, B. 2016. Measuring the Distance Between Finite Markov Decision Processes. In Jonker, C. M.; Marsella, S.; Thangarajah, J.; and Tuyls, K., eds., *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 468–476. ACM.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. The MIT Press, Cambridge.

Taitler, A.; Gimelfarb, M.; Gopalakrishnan, S.; Liu, X.; and Sanner, S. 2022. pyRDDLGym: From RDDL to Gym Environments. *CoRR*, abs/2211.05939.

Taylor, M. E.; and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(56): 1633–1685.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J. B.; and Isola, P. 2020. Rethinking Few-Shot Image Classification: A Good Embedding is All You Need? In *ECCV (14)*, volume 12359 of *Lecture Notes in Computer Science*, 266–282. Springer.

van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR*, abs/1509.06461.

Vecerík, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. A. 2017. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *CoRR*, abs/1707.08817.

Vettoruzzo, A.; Bouguelia, M.; Vanschoren, J.; Rögnvaldsson, T. S.; and Santosh, K. 2023. Advances and Challenges in Meta-Learning: A Technical Review. *CoRR*, abs/2307.04722.

Wang, Z.; and Carbonell, J. G. 2018. Towards More Reliable Transfer Learning. In *ECML/PKDD (2)*, volume 11052 of *Lecture Notes in Computer Science*, 794–810. Springer.

Yeh, W. W.-G. 1985. Reservoir management and operations models: A state-of-the-art review. *Water resources research*, 21(12): 1797–1818.

Zamani, Z.; Sanner, S.; and Fang, C. 2012. Symbolic Dynamic Programming for Continuous State and Action MDPs. In *AAAI*. Toronto, Canada.

Zhang, W.; Deng, L.; Zhang, L.; and Wu, D. 2023. A Survey on Negative Transfer. *IEEE CAA J. Autom. Sinica*, 10(2): 305–329.

Zhu, Z.; Lin, K.; and Zhou, J. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *CoRR*, abs/2009.07888.