Interactive Reinforcement Learning with Dynamic Reuse of Prior Knowledge from Human and Agent Demonstrations

Zhaodong Wang and **Matthew E. Taylor**
School of EECS, Washington State University
{zhaodong.wang, matthew.e.taylor}@wsu.edu

Abstract

Reinforcement learning has enjoyed multiple impressive successes in recent years. However, these successes typically require very large amounts of data before an agent achieves acceptable performance. This paper focuses on a novel way of combating such requirements by leveraging existing (human or agent) knowledge. In particular, this paper leverages demonstrations, allowing an agent to quickly achieve high performance. This paper introduces the Dynamic Reuse of Prior (DRoP) algorithm, which combines the offline knowledge (demonstrations recorded before learning) with an online confidence-based performance analysis. DRoP leverages the demonstrator's knowledge by automatically balancing between reusing the prior knowledge and the current learned policy, allowing the agent to outperform the original demonstrations. We compare with multiple state-of-the-art learning algorithms and empirically show that DRoP can achieve superior performance in two domains. Additionally, we show that this confidence measure can be used to selectively request additional demonstrations, significantly improving the learning performance of the agent.

1 Introduction

There have been increasingly successful applications of reinforcement learning [Sutton and Barto, 1998] (RL) methods in both virtual agents and physical robots. However, RL often suffers from slow learning speeds in complex domains, which is particularly detrimental when initial performance is critical. External knowledge may be leveraged by RL agents to improve learning — demonstrations have been shown to be useful for many types of agents' learning [Schaal, 1997; Argall *et al.*, 2009]. In contrast to many *behavior cloning* methods, which seek to mimic the demonstrated behavior, our goal is to leverage a demonstration to learn faster, and ultimately outperform the demonstrator.

Inverse reinforcement learning (IRL) [Ng *et al.*, 2000] is an alternative to behavior cloning where the agent aims to estimate the demonstrator's reward function and then optimize it. It is typically used in cases where no environmental reward is available, and often requires the transition model.

To further improve over the demonstrations, one approach is the Human Agent Transfer [Taylor *et al.*, 2011] (HAT) algorithm, which formulates the problem as that of transfer learning [Taylor and Stone, 2009]: a source agent can demonstrate a policy and then a target agent can improve its performance over that policy. As refinement, the Confidence Human Agent Transfer [Wang and Taylor, 2017] algorithm was proposed by leveraging the confidence in a policy.

In order to leverage demonstrations to improve learning, four problems must be considered. First, the demonstration may be suboptimal, and the agent should aim to improve upon it. Second, if there are multiple demonstrators, their outputs must be combined in a way to handle any inconsistencies [Mao *et al.*, 2018]. Third, the demonstration is rarely exhaustive and some type of generalization must be used to handle unseen states. Fourth, the agent must balance the usage of the prior knowledge and its own self-learned policy.

In this paper, we introduce DRoP (Dynamic Reuse of Prior) as a interactive method to assist RL by addressing the above problems. Prior research [Chernova and Veloso, 2007; Wang and Taylor, 2017] used offline confidence. In contrast, DRoP leverages temporal difference models to achieve online confidence-based performance measurement on transferred knowledge for better domain adaption. To guarantee convergence, we introduce an action selection method to help the target agent balance between following the demonstration and following its own learned knowledge. We empirically evaluate DRoP using the domains of Cartpole and Mario, showing improvement over existing methods, compared with other state-of-art demonstration learning methods. Results also validate our claim that multiple experts' demonstrations can be leveraged simultaneously, and that DRoP is able to distinguish between high- and low-quality demonstrations automatically. Finally, we show that these confidence measures can be used to actively request additional demonstrations, significantly improving learning performance.

The main contributions of this paper are: 1) automatically balancing between an existing demonstration and a self-learned policy, 2) efficiently integrating demonstrations from multiple sources by distinguishing the knowledge quality, and 3) actively requesting demonstrations in low confidence states.

2 Background

This section presents a selection of relevant background knowledge and techniques from recent research.

2.1 Reinforcement Learning

By interacting with an environment, an RL agent can learn a policy to maximize an external reward. A Markov decision process is common formulation of the RL problem. In a Markov decision process, A is a set of actions an agent can take and S is a set of states. There are two (initially unknown) functions within this process: a transition function ($T : S \times A \mapsto S$) and a reward function ($R : S \times A \mapsto \mathbb{R}$).

The goal of an RL agent is to maximize the expected reward — different RL algorithms have different ways of approaching this goal. This paper uses Q-learning [Watkins and Dayan, 1992] as the base RL althorithm:

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

2.2 Transfer Learning and Learning from Demonstration

The key idea of transfer learning is to leverage existing knowledge to improve a new agent's learning performance. Transfer learning has been applied in various domains, such as multitask learning [Kirkpatrick *et al.*, 2017; Teh *et al.*, 2017], deep reinforcement learning [Rusu *et al.*, 2016; Parisotto *et al.*, 2016; Higgins *et al.*, 2017], and representation learning [Maurer *et al.*, 2016; Luo *et al.*, 2017]. In this section, we will discuss knowledge transfer techniques using demonstrations.

Probabilistic Policy Reuse [Fernández and Veloso, 2006] is one transfer learning approach. Like many other existing approaches, it assumes both the source and the target agents share the same internal representations and optimal demonstrations are required. Existing policies could guide the learning direction as shown elsewhere [Da Silva and Mackworth, 2010; Brys *et al.*, 2017], but near-optimal policies could be impracticable due to the complexity of the learning task or the cost of a domain expert's time.

Imitation is a popular and fundamental approach that transfers the demonstrator's behavior by having an agent exactly following the demonstrations. However the learner's performance could possibly be limited by the demonstrator. On top of imitation learning, Dagger [Ross *et al.*, 2011] incorporates the demonstration trajectories into the target agent's real visited states. Dagger works by collecting a dataset (real visited state-action pairs) at each iteration under the current policy (starting from the pure demonstration policy) and trains the next policy under the aggregate of all collected datasets. It shows the ability of adapting demonstrations into the target agent's own learning experience and outperforming the pure demonstration reuse. However, the shortcoming is that Dagger treats the policy training simply as a supervised learning problem and the only contribution is how it organizes the training data set for each iteration of building the action classifier. This would limit the adaptive capacity of Dagger, particularly when the target task's state environment (transition MDPs) is not exactly the same as the demonstration's.

Human Agent Transfer (HAT) takes a novel step by integrating the demonstrations with RL. The goal of HAT is to leverage demonstration from a source human or source agent, and then improve agents' performance with RL. *Rule transfer* [Taylor and Stone, 2007] is used in HAT to remove the requirements on sharing the same internal representation between source and target agents, which is the novel approach that allows knowledge transfer across different types agents (e.g., from human to an agent). The following steps summarize HAT:

1. Learn a policy ($\pi : S \mapsto A$) from the source task.
2. Train a decision list upon the learned policy as "IF-ELSE" rules.
3. The target agent's action is guided by the trained rules under a decaying probability.

By fitting the demonstration reuse into RL domains with reward distributions, HAT can better adapt the transferred policy into the target task comparing to the pure classifier training of Dagger.

As an extension, Confidence Human Agent Transfer [Wang and Taylor, 2017] (CHAT) provides a method based on confidence — it leverages a confidence-based source agent's/human's demonstration to improve the learning performance. Instead of rule transfer, CHAT measures the confidence in the source demonstration. Such offline confidence is used to predict how reliable the transferred knowledge is. To assist RL, CHAT will leverage the source demonstrations to suggest an action in the agent's current state, along with the calculated confidence. For example, CHAT can use a Gaussian distribution to predict an action from a demonstration with an offline probability. If the calculated confidence is higher than a pre-tuned confidence threshold, the agent would consider the prior knowledge reliable and execute the suggested action.

To guarantee that the demonstration data will not harm the agent's learning convergence, all above methods use similar solutions — following the artificial probability control, which forces the agent into reusing the prior knowledge under a decaying probability curve.

3 Dynamic Reuse of Prior (DRoP)

This section introduces DRoP, a method to estimate the confidence of the agent in different data sources over time. Section 3.1 discusses how we build the confidence-based policy from the demonstration dataset. Section 3.2 introduces the confidence update methods which dynamically analyze the prior confidence during online learning. Based on the confidence update, Section 3.3 shows how an agent should select an action to execute, as well as how multiple sources could be integrated by DRoP.

DRoP follows a three step process:

1. Collect a demonstration dataset (state-action pairs).
2. Use supervised learning to train a policy on the demonstration data. Different methods (e.g. IRL, HAT, etc.) could be applied in this step but this paper uses two methods: Gaussian regression and a fully connected

neural network. The demonstration data might not be i.i.d. for supervised learning and hence the policy might be sub-optimal. We will show in the results that our method can still improve the performance without an optimal start policy.

3. DRoP's bootstrap (Algorithm 1) is used to assist an RL agent in the target task. This is the core step that helps the agent tranfer the prior to its own knowledge base during self-learning.

Relative to other existing work, there are significant advantages to DRoP's online confidence measurement: First, it removes the trial-and-error confidence threshold tuning process. Second, the target agent's experience is used to measure confidence on demonstrations. DRoP performs the adaptive confidence-based performance analysis during the target agent's learning. This online process can help guarantee the transfer knowledge is adapted to the target tasks. Third, there is no global reuse probability control, a parameter that is crucial in other knowledge reuse methods [Wang and Taylor, 2017; Taylor *et al.*, 2011; Fernández and Veloso, 2006] to avoid suboptimal asymptotic performance.

3.1 Policy Models Using Confidence-based Classification

A **Gaussian** could be an effective kernel for the policy regression because we consider the demonstrations recorded from human teachers, which could contain noise. Given the demonstration dataset $D = \{x_1, a_1, x_2, a_2, ..., x_i, a_i\}$, $a \in A$, and a query state point x, the policy model would make an action prediction using the Gaussian kernel $K(x, x_i)$:

$$f(x) = \arg\max_{a \in A} \sum_{x_i \in D,\ a_i = a} K(x, x_i) \ / \ N_a$$

normalized by total number of action a in the dataset: $N_a = \#\{a_i \in A \mid a_i = a\}$, where A is the set of all possible actions. The confidence of the above prediction is computed through the weighted probability:

$$P = \frac{\sum_{x_i \in D,\ a_i = a} K(x, x_i) \ / \ N_a}{\sum_{\hat{a} \in A} \sum_{x_i \in D,\ a_i = \hat{a}} K(x, x_i) \ / \ N_{\hat{a}}}$$

The other policy model considered is a two-hidden-layer **neural network** (NN). The prediction with the highest confidence is made through a softmax [Bishop, 2006] layer:

$$P = \max \left\{ \frac{1}{\sum_i exp(\theta_i^T \cdot p)} \begin{bmatrix} exp(\theta_1^T \cdot p)) \\ exp(\theta_2^T \cdot p)) \\ ... \\ exp(\theta_i^T \cdot p)) \end{bmatrix} \right\}$$

θ_i is the weight vector of the i-th output of the softmax layer and p is the corresponding input. $\max\{\cdot\}$ in the above equation is the weighted confidence by the network.

3.2 Temporal Difference Confidence Update

The online confidence metric is measured via a temporal difference (TD) approach. For each action source (learned Q function or prior knowledge), we build a TD model to measure the confidence-based performance via experience.

A confidence-based TD model is used to analyze the performance level of every action source with respect to every state. Once an action is taken, the confidence model will update the corresponding action source's confidence value. As a heuristic, an RL agent should prefer the action source with higher confidence value: the expected reward would likely be higher by taking the action from that source.

Our TD confidence model, $C(s)$, updates as follows:

$$C(s) \leftarrow (1 - F(\alpha)) \times C(s) + F(\alpha) \times [G(r) + \gamma \times C(s')] \quad (1)$$

where γ is discount factor, r is original task reward, α is the update parameter, and $F(\alpha)$ and $G(r)$ are defined below. For continuous domains, function approximators such as tile coding [Albus, 1981] should be used — in this work we are using the same discretization approximator as $Q(s, a)$.

We have two types of $C(s)$: confidence prior knowledge model ($CP(s)$) and confidence Q knowledge model ($CQ(s)$). $CP(s)$ denotes the confidence of following the prior knowledge and $CQ(s)$ denotes the confidence following self-learned policy, given the current state s. $CP(s)$ has 2 update methods: Dynamic Rate Update (DRU) and Dynamic Confidence Update (DCU). For DRU, we define a dynamic updating rate based on classification confidence distribution discussed in Section 3.1: $F(\alpha) = \alpha \times P$. The update rate of $CP(s)$ will be bounded by the confidence of the corresponding classification. If the confidence is higher, the update step will be more confident with a higher update rate (and vice versa). Besides, we use the original task reward: $G(r) = r$.

For DCU, we use a fixed update rate: $F(\alpha) = \alpha$, but the reward function leverages the confidence: $G(r) = \frac{r}{r\_max} \times P$, where $\frac{r}{r\_max}$ is a normalized reward (r_max denotes the maximum absolute reward value) and G(r) re-scales the reward using the confidence distribution.

$CQ(s)$ uses the same update methods with $F(\alpha) = \alpha$ and $G(r) = r$. $CQ(s)$ will be updated only if an action is selected by the agent's own Q knowledge (from $Q(s, a)$) and $CP(s)$ will be updated only if an action is selected from the prior.

3.3 Action Source Selection Method

Given these TD-based confidence models, we define our action selection methods to balance an agent's learned knowledge ($CQ(s)$) with its prior knowledge ($CP(s)$).

Intuitively, higher confidence is better and the **hard decision** defines action source (AS) as:

$$AS = \arg\max[\{CQ(s), CP(s)\}],$$

where ties are broken randomly.

However, lower-confidence action might be worth trying due to the imperfection of the prior model training. The **soft decision** decides the action source using a probability distribution. To calculate the probability, we first normalize $CQ(s)$ and $CP(s)$: $R = \max\{|CQ(s)|, |CP(s)|\}$, $rCQ = CQ(s)/R$, $rCP = CP(s)/R$. Then we rescale rCQ and rCP using the hyperbolic tangent function: $tanh(\cdot)$. The probability of selecting action source is defined as:

Algorithm 1: DRoP, Target Learning Bootstrap

Input: Prior knowledge model PM
for *each episode* **do**
  Initialize state s to start state
  for *each step of an episode* **do**
    if *rand()* $\leq \epsilon$ **then**
      $a \leftarrow$ random action ▷ Exploration
    else
      $AS \leftarrow$ Action Decision Model ▷ S-H-ϵ
      if $AS ==$ *Prior Knowledge* **then**
        $a \leftarrow$ action from Prior Knowledge
      else
        $a \leftarrow$ action that maximizes Q
    Execute action a
    Observe new state s' and reward r
    if $AS ==$ *Prior Knowledge* **then**
      Update CP as Equation 1
    else
      Update CQ as Equation 1
    Update Q (SARSA, Q-Learning, etc.);

$$AS = \begin{cases} Q & P = \frac{\tanh(rCQ)+1}{\tanh(rCP)+\tanh(rCQ)+2} \\ Prior & P = \frac{\tanh(rCP)+1}{\tanh(rCP)+\tanh(rCQ)+2} \end{cases} \quad (2)$$

The **soft-hard-ϵ decision** (S-H-ϵ), shown in Algorithm 2, takes advantage of the above two models by adding an ϵ-greedy switch over the hard decision and the soft decision: S-H-ϵ can both greedily exploit the confidence value and also perform probabilistic exploration. S-H-ϵ is used in the result section because it outperforms the other two models.

When there are **multiple sources** of prior knowledge, the above AS (in Equation 2) can be expanded to multiple cases:

$$AS = \begin{cases} Prior_1 & P_1 = \frac{\tanh(rCP_1)+1}{\sum_i \{\tanh(rCP_i)+1\}} \\ \quad ... & \quad ... \\ Prior_i & P_i = \frac{\tanh(rCP_i)+1}{\sum_i \{\tanh(rCP_i)+1\}} \end{cases} \quad (3)$$

3.4 Interactive Demonstration Request

It may be useful to request additional demonstrations where the agent can most benefit from them. These regions would be difficult to pre-compute, but they can be identified during training as places in the state space where the agent has low confidence in its policy. We first record demonstrations from an human expert and use DRoP to assist an RL agent's learning. After a short period of training (2% of the baseline training time), we then use the following steps to request additional demonstrations within the same number of episodes from the same demonstrator at each state s:

1. Calculate average confidence of prior knowledge (i.e., $CP(s)$) at each step of the current episode:

$$AveC = \frac{1}{steps} \times \sum_i CP(s_i)$$

Algorithm 2: S-H-ϵ, Soft-Hard-ϵ Decision Model

Input: CQ, CP, State s
$R = \max\{|CQ(s)|, |PQ(s)|\}$
$rCQ = CQ(s)/R$
$rCP = CP(s)/R$
if *rand()* $\leq \epsilon$ **then**
  if *rand()* $\leq \frac{\tanh(rCQ)+1}{\tanh(rCP)+\tanh(rCQ)+2}$ **then**
    AS = Prior Knowledge
  else
    AS = Q Knowledge
else
  $AS = \arg\max[\{CQ(s), CP(s)\}]$
return AS ▷ Action source

2. Scan a neighbourhood window of states and calculate the average "$CP(s)$" to approximate the local confidence. An average is used to reduce noise.
3. If the averaged CP value is smaller than $AveC$, request a demonstration of 20 actions, starting at the current state.
4. Add the above recorded state-action pairs into the request demonstration dataset of DRoP.

Section 5.3 will show that the requested demonstrations would help the learning more than the original recorded data.

4 Experimental Setup

This section details our experimental methodology.

4.1 Experiment Domains

We evaluate our method in two domains: Cartpole and Mario.

The **Cartpole** simulation is based on the open-source OpenAI Gym [Brockman *et al.*, 2016]. This task has a continuous state space; the world state is represented as 4-tuple vector: position of the cart, angle of the pole, and their corresponding velocity variables. The system is controlled by applying a force of +1 or -1 to the cart. The reward function is designed as $+1$ for every surviving step and -500 if the pole falls.

Mario is a benchmark domain [Karakovskiy and Togelius, 2012] based on Nintendo's Mario Brothers. To guarantee the diversity and complexity of tasks, our simulation world is randomly sampled from a group of one million worlds. The world state is represented as a 27-tuple vector, encoding the agent's state/position information, surrounding blocks, and enemies [Suay *et al.*, 2016]. There are 12 ($3 \times 2 \times 2$) actions (movement direction $\times$ jump button $\times$ Run/Fire button).

4.2 Methodology

Demonstrations are collected either from a human participant (one of the authors of this paper) via a simulation visualizer, or directly from an agent executing the task.

We use a "4-15-15-2" network (15 nodes in two hidden layers) network in Cartpole and a "27-50-50-12" network in Mario. To benchmark against CHAT, we use the same networks as the confidence models used by DRoP. To benchmark against HAT, J48 [Quinlan, 1993] is used to train decision rules. For both CHAT and HAT, the self-decaying

reuse probability control parameter Φ was tuned to be 0.999 in Cartpole and 0.9999 in Mario. For Dagger, we use the best parameters of "D0.5"[1]. Target agents in both Cartpole and Mario use the Q-learning algorithm. In Cartpole, we use $\alpha = 0.2$, $\gamma = 0.9$, and $\epsilon = 0.1$. In Mario, we use $\alpha = \frac{1}{10\times32}$, $\gamma = 0.9$, and $\epsilon = 0.1$. These parameters are set to be consistent with previous research [Brys *et al.*, 2015; Wang and Taylor, 2017] in these domains.

Experiments are evaluated in terms of learning curves, the jumpstart, the total reward, and the final reward. Jumpstart is defined as the initial performance improvement, compared to an RL agent with no prior knowledge. The total reward accumulates scores every 5 percent of the whole training time. Experiments are averaged over 10 learning trials and t-tests are performed to evaluate the significance. Error bars on the learning curves show the standard deviation.[2]

5 Experimental Results

This section will present and discuss our experimental results. We first show the improvement over existing knowledge reuse algorithms, HAT, Dagger and CHAT, as well as baseline learning. Then we show DRoP is capable of leveraging different quality demonstrations from multiple sources. Finally we will evaluate how DRoP could be used for interactive RL by involving a human demonstrator in the loop.

5.1 Improvement over Baselines

In Cartpole, we first let a trained agent demonstrate 20 episodes (average number of steps: 821 ± 105) and record those state-action pairs. In Mario, we let a trained agent demonstrate 20 episodes (average reward: 1512 ± 217).

DRoP is then used with these demonstration datasets. As benchmarks, we run HAT, CHAT, and Dagger on the same datasets, and Q-learning is run without prior knowledge. Learning performance is compared in Table 1. DRoP with different models outperforms the baselines. The top two scores for each type of performance metrics are in bold. DRoP with DRU and S-H-ϵ model has achieved the best learning result and further experiments in the next sections use this setting. DRoP's improvements are all statistically significant ($p < 10^{-4}$) relative to Q-learning and we mark results with no significant improvement ($p > 0.05$) of other benchmark methods in italics. To highlight the improvement, Figure 1 and 2 show the learning curves of DRoP using neural network (NN) policy model and DRU method. DRoP outperforms all baselines and all 4 ways of DRoP work (i.e. DRU/DCU and NN/Gaussian).

Methods like CHAT have to use a global self-decaying probabilistic parameter to control the reuse frequency of prior knowledge. Therefore, it is possible that the target agent will execute suboptimal actions repeatedly (subject to the decaying probability). From Figure 1 and 3, we can see that as the reuse probability decays, the performance of CHAT dips, suggesting the agent must re-explore and re-learn to outperform the previously learned (suboptimal) knowledge. In contrast, DRoP allows the target agent to always perform online confidence-based performance analysis, which would achieve a better balance between the prior and self-learned policy.

[1]For each episode i, the reuse probability is 0.5^{i-1}.

[2]Our sources are available at zhaodongwang.info

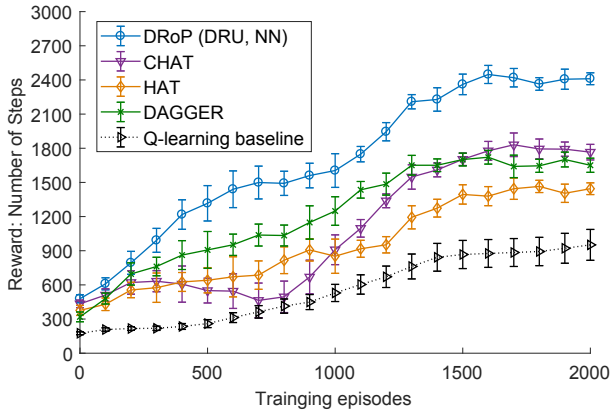

Figure 1: Comparison of learning rewards in Cartpole.

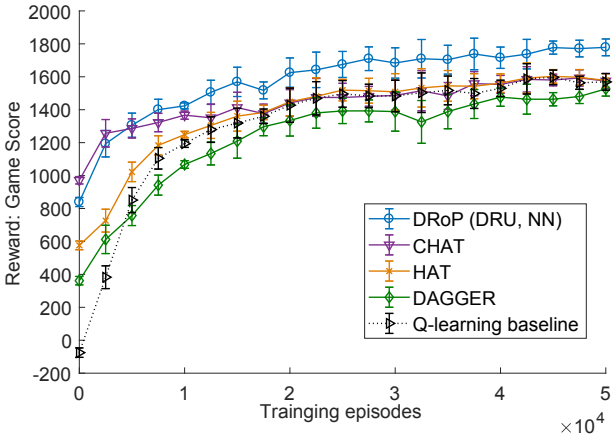

Figure 2: Comparison of learning rewards in Mario.

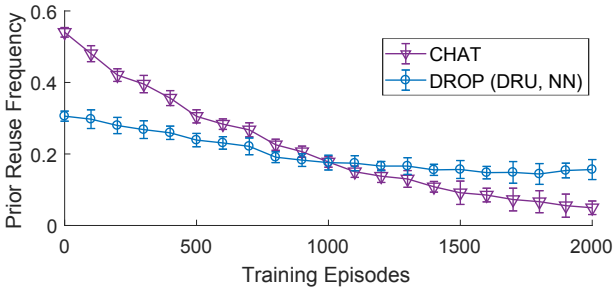

Figure 3: Comparison of actual reuse frequency of prior knowledge using DRoP and CHAT in Cartpole.

5.2 DRoPping Low-quality Demonstrations

We consider using suboptimal demonstrations to see how well the online confidence-based analysis mechanism can handle poor data without harming the optimal convergence. Here we have five groups of demonstrations (recorded from different agents), ranging from completely random to high performing (Rand to L4, shown in Tables 2 and 3).

Method	Cartpole			Mario		
	Jumpstart	Total Reward	Final Reward	jumpstart	Total Reward	Final Reward
Q-Learning	N/A	11653	951 ± 36	N/A	27141	1569 ± 51
HAT	201	20004	1444 ± 52	651	28828	*1577* ± 41
CHAT	258	22692	1766 ± 68	**1046**	30144	*1574* ± 46
Dagger	144	25728	1651 ± 62	*437*	*25828*	*1526* ± 44
DRoP (DCU, Gaussian)	**315**	34025	2369 ± 72	923	**32562**	1699 ± 49
DRoP (DRU, Gaussian)	**320**	35187	**2395 ± 64**	**931**	31241	1732 ± 46
DRoP (DCU, NN)	308	**35312**	2383 ± 71	909	32108	**1752 ± 55**
DRoP (DRU, NN)	303	**35544**	**2411 ± 56**	915	**33022**	**1779 ± 61**

Table 1: This table compares baselines (methods 1 to 4) with DRoP using different models (methods 5 to 8). Jumpstart, total reward, and final reward are shown. The top two scores of each column are in bold and insignificant improvements over Q-learning are in italics.

Demo Performance	Jump start	Converged Performance	Converged Reuse Frequency
Q-Learning	N/A	951 ± 136	N/A
Rand: 15 ± 7	-5	942 ± 142	0.02 ± 0.01
L1: 217 ± 86	153	1453 ± 96	0.12 ± 0.03
L2: 435 ± 83	211	1765 ± 112	0.17 ± 0.04
L3: 613 ± 96	278	2080 ± 86	0.21 ± 0.02
L4: 821 ± 105	303	2411 ± 56	0.32 ± 0.03

Table 2: This table shows the performance of Q-learning and DRoP (DRU, NN) upon 5 different levels of demonstrations in Cartpole.

Demo Performance	Jump-start	Converged Performance	Converged Reuse Frequency
Q-Learning	N/A	1569 ± 51	N/A
Rand: -245 ± 11	-52	1552 ± 72	0.01 ± 0.01
L1: 315 ± 183	336	1582 ± 67	0.08 ± 0.02
L2: 761 ± 195	512	1601 ± 73	0.15 ± 0.05
L3: 1102 ± 225	784	1695 ± 81	0.19 ± 0.03
L4: 1512 ± 217	906	1779 ± 61	0.28 ± 0.04

Table 3: This table shows the performance of Q-learning and DRoP (DRU, NN) upon 5 different levels of demonstrations in Mario.

We first evaluate our method *individually* with the five demonstration datasets. Cartpole results are shown in Table 2 and Mario results are shown in Table 3. As we can see, the quality of the demonstration does effect performance, and better demonstrations lead to better performance. However, what is more important is whether poor demonstrations hurt learning. If we look at the results of using randomly generated demonstrations, we find that even if the jumpstart is negative (i.e., the initial performance is hurt by using poor demonstrations), the final converged performance is almost the same as learning without the poor demonstrations, which means the DRoP agent has learned to ignore the poor demonstrations (reuse frequency is almost zero).

We then evaluate the multiple-case model (Equation 3) by *simultaneously* providing the above demonstrations to DRoP and results are shown in Table 4. With bad demonstrations mixed in the data, DRoP is still able to reuse L4 the most, correctly leveraging the highest-quality data.

5.3 DRoP-in Requests for Demonstrations

This section addresses another challenging problem — can DRoP's confidence mechanism productively request additional demonstrations from a human or agent?

Method	Jumpstart	Converged Performance	Converged Reuse Frequency
CHAT	191	983 ± 151	0.05 ± 0.02
DRoP	253	2286 ± 91	Rand: 0.02 ± 0.01 L1: 0.05 ± 0.01 L2: 0.06 ± 0.02 L3: 0.11 ± 0.03 L4: 0.23 ± 0.02

Table 4: This table shows the performance of DRoP (DRU, NN) and CHAT upon multiple sources of demonstrations in Cartpole.

In Mario, we first record 20 episodes of demonstrations from an human expert with an average score of 1735, and then use the method in Section 3.4 for active demonstrations requests. We find that a window of 10×10 neighbourhood positions works well and this could be domain-variant. The time cost (348 s) of the active demonstration collection (20 episodes) is only 2% of the baseline training time (15325 s), highlighting the efficiency of DRoP.

Figure 4 shows the performance comparison between the above two demonstration datasets: 20 episodes of original human demonstrations and 20 episodes requested by DRoP. Notice that even though human's demonstration performance is higher than the L4 dataset from the previous section, the actual jumpstart of the former is instead lower. This is potential evidence that the agent could not "digest" the entire human demonstrator's knowledge. DRoP would request the demonstration from human only in states where the knowledge confidence is relatively low. Therefore, we know that the target agent truly needs these requested demonstrations and the learning improvement could be better. DRoP improved the overall learning effectiveness by requesting less, but critical, demonstration data.

6 Conclusion and Future Work

This paper has introduced DRoP and evaluated it in two domains. This work shows that by integrating offline confidence with online temporal difference analysis, knowledge transfer from source agents or humans can be successfully achieved. DRoP outperformed both learning without prior knowledge and recent state-of-the-art transfer methods.

DRoP works as an action communication protocol between

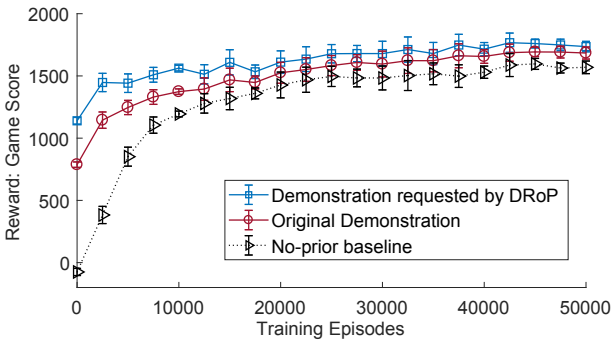

Figure 4: Mario learning curves using demonstration requested by DRoP and original demonstration from human expert.

the transferred offline prior knowledge and online reinforcement learning, leveraging the TD confidence updates. Results have shown that DRoP can effectively leverage the prior knowledge of different quality levels from multiple sources. Besides, demonstrations requested by DRoP can significantly improve the RL agent's learning process, leading to a more efficient collaboration between two very different types of knowledge entities: humans and agents.

There are a number of interesting future directions, including the following. First, we will explore model-based demonstrations to see if any hierarchical structure could provide better confidence measurement than classification equally over all state-action pairs. Second, we will use DRoP to build a lifelong online learning system which can automatically refine that knowledge model during learning. Third, considering the high training cost of deep learning, DRoP could be implemented as a bootstrap leveraging light-weight pre-trained prior knowledge network from demonstrations.

References

[Albus, 1981] JS Albus. Brains, behavior. *& Robotics. Peterboro, NH: Byte Books*, 1981.

[Argall *et al.*, 2009] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[Bishop, 2006] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Brys *et al.*, 2015] Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 181–188, 2015.

[Brys *et al.*, 2017] Tim Brys, Anna Harutyunyan, Peter Vrancx, Ann Nowé, and Matthew E Taylor. Multi-objectivization and ensembles of shapings in reinforcement learning. *Neurocomputing*, 2017.

[Chernova and Veloso, 2007] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[Da Silva and Mackworth, 2010] Bruno N Da Silva and Alan Mackworth. Using spatial hints to improve policy reuse in a reinforcement learning agent. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 317–324, 2010.

[Fernández and Veloso, 2006] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 720–727. ACM, 2006.

[Higgins *et al.*, 2017] Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Thirty-fourth International Conference on Machine Learning*, 2017.

[Karakovskiy and Togelius, 2012] Sergey Karakovskiy and Julian Togelius. The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, 2012.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[Luo *et al.*, 2017] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*, pages 164–176, 2017.

[Mao *et al.*, 2018] Li Mao, Wei Yi, and Kudenko Daniel. In *Adaptive and Learning Agents Workshop at AAMAS-18*, 2018.

[Maurer *et al.*, 2016] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884, 2016.

[Ng *et al.*, 2000] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

[Parisotto *et al.*, 2016] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

[Quinlan, 1993] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Ross *et al.*, 2011] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[Rusu *et al.*, 2016] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[Schaal, 1997] Stefan Schaal. Learning from demonstration. In *Advances in neural information processing systems*, pages 1040–1046, 1997.

[Suay *et al.*, 2016] Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 429–437, 2016.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Taylor and Stone, 2007] Matthew E Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 879–886. ACM, 2007.

[Taylor and Stone, 2009] Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[Taylor *et al.*, 2011] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, May 2011.

[Teh *et al.*, 2017] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4499–4509, 2017.

[Wang and Taylor, 2017] Zhaodong Wang and Matthew E. Taylor. Improving Reinforcement Learning with Confidence-Based Demonstrations. In *Proceedings of the 26th International Conference on Artificial Intelligence (IJCAI)*, August 2017.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.