Artificial Intelligence Review, Volume 55, Issue 4, April 2022, pp. 3213-3241
DOI:10.1007/s10462-021-10085-1

Model-Free Reinforcement Learning from Expert Demonstrations: A Survey

Jorge Ramírez · Wen Yu · Adolfo Perrusquía

Received: date / Accepted: date

Abstract Reinforcement learning from expert demonstrations (RLED) is the intersection of imitation learning with reinforcement learning that seeks to take advantage of these two learning approaches. RLED uses demonstration trajectories to improve sample efficiency in high-dimensional spaces. RLED is a new promising approach to behavioral learning through demonstrations from an expert teacher. RLED considers two possible knowledge sources to guide the reinforcement learning process: prior knowledge and online knowledge.

This survey focuses on novel methods for model-free reinforcement learning guided through demonstrations, commonly but not necessarily provided by humans. The methods are analyzed and classified according to the impact of the demonstrations. Challenges, applications, and promising approaches to improve the discussed methods are also discussed.

Keywords: reinforcement learning, imitation learning, learning from demonstrations, behavioral learning, demonstrations.

1 Introduction

One of the artificial intelligence's most famous applications is AlphaGo [79], an agent capable of playing Go professionally. It trains with a network of super-

Jorge Ramírez
Departamento de Control Automático, CINVESTAV-IPN (National Polytechnic Institute), Mexico City, Mexico

Wen Yu
Departamento de Control Automático, CINVESTAV-IPN (National Polytechnic Institute), Mexico City, Mexico
The corresponding author is Wen Yu.
E-mail: yuw@ctrl.cinvestav.mx

Adolfo Perrusquía
School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, United Kingdom.

Published by Springer. This is the Author Accepted Manuscript issued with: Creative Commons Attribution Non-Commercial License (CC:BY:NC 4.0).
The final published version (version of record) is available online at DOI:10.1007/s10462-021-10085-1. Please refer to any applicable publisher terms of use.

vised learning policies based on expert human demonstrations, and then reinforcement learning is used to fine-tune supervised learning policies. Similarly, AlphaStar [92], an agent capable of playing StarCraft II with a Grandmaster level, is initially trained using supervised learning with human demonstrations and later enhanced using reinforcement learning. AlphaGo and AlphaStar share something in common, the behavioral learning approach used from human demonstrations, which dramatically affect agent performance.

The most common approach to behavioral learning is Imitation Learning (IL). Generally, IL is more concerned with observations than with actions. Learning from Demonstration (LfD) [2] [69] and Programming by Demonstration (PbD) [8] are other names of IL commonly used in the robotics field, while IL is the name most widely used in the field of machine learning. These three terms are used to describe the process of learning through the use of demonstration. There is no real distinction in IL, LfD, and PbD.

IL began to be used in robotics manufacturing to program robot movements [78][57]. Currently, IL has gained popularity using machine learning techniques such as supervised learning, which achieves local generalization [7]. In IL, a subject called an apprentice learns to mimic a teacher's behavior in performing a task, the teacher, who can be considered an expert, provides demonstrations of the desired behavior by performing tasks. It is important to note that in IL, the teacher's behavior is supposed to be optimal, which is not entirely accurate since the teacher, who can be a human, naturally makes mistakes and could follow a lousy strategy in demonstrating a task. It is, therefore, more accurate to assume the teacher's behavior as suboptimal. Since the amount of data provided by demonstrations is relatively small for today's computing capabilities, the most significant IL advantages are its practicality and sample efficiency.

There are four different methods to perform demonstrations, according to [2][44][69]:

- Teleoperation: in this method, the teacher operates the apprentice through some input device, like a joystick or a haptic device.
- Kinesthetic teaching: in this method, the teacher manually moves the apprentice's body, which is useful when the apprentice has a body easy to move for an average human.
- Sensors mounted on the teacher: in this method, the sensors are mounted directly on the teacher's execution platform.
- External observation: in this method, sensors are not mounted on the apprentice's body and the teacher's execution platform. Artificial vision techniques are applied here.

Another common approach to behavior learning is Reinforcement Learning (RL) [85]. This approach, unlike IL, learns a behavior that achieves a global generalization to solve a problem in the most optimal way through trial and error, exploring and exploiting states and actions through a criterion that rewards or punishes an agent depending on the agent's performance with respect to the desired behavior. In general, this approach is not sample-efficient, as a

large amount of data is required to learn an optimal control policy. This approach has become popular in recent years due to its excellent generalization capabilities. The help of advances in deep learning and increased computational capacity has led to increasingly efficient RL algorithms. This approach has shown to be a powerful tool in resolving more and more complex real-world problems. The evolution of RL using deep learning techniques is called Deep Reinforcement Learning (DRL). RL is a simple computational method based on Dynamic Programming (DP) [5] [84]. With RL, a control policy is obtained by progressively learning an optimal solution online, with partial or no knowledge of the dynamic environment model [51].

Reinforcement Learning from Expert Demonstrations (RLED) is a new approach to behavioral learning, which is the intersection between IL and RL. RLED tries to benefit from both approaches' advantages while trying to avoid their disadvantages. RLED seeks to learn an optimal behavior, capable of generalizing globally and also being sample-efficient. RLED is intended to be a practical approach in real-world automatic control problems, mainly capable of solving challenging problems that IL cannot solve and where RL alone would take an unacceptable amount of time to solve.

The RLED approach uses demonstrations to improve sample efficiency in high-dimensional spaces, helping when sparse rewards are used as reward specification and with the possibility of safe exploration. The first to propose an intersection of RL and IL was Schaal in [74], which shows how learning from demonstrations is beneficial for the RL process.

RLED is similar to other approaches such as Inverse RL [63], where knowledge is available through demonstrations of an expert teacher but differs in that a reward function is not available. Instead, the reward function is inferred from the demonstration trajectories. Another similar approach is Batch RL [48], also known as Offline RL [50], where only prior knowledge is available, and the RL agent does not have access to interactions with the environment. The agent should use the limited source of prior knowledge to learn the best policy it can.

There have been many impressive advances in RLED. With constant domain growth, a survey needs to organize and clearly define its shared ideas. Therefore, this survey paper offers an overview of the most relevant novel methods for model-free Reinforcement Learning guided through expert demonstrations, commonly but not necessarily from humans, where we assume knowledge (leveraged by the demonstrator) is available. The contributions of this survey paper are:

- The three general assumptions made by these particular kinds of algorithms are established.
- Relevant RLED algorithms are analyzed and classified according to how demonstrations are applied.
- Current applications, challenges, and a promising approach to improving RLED algorithms are analyzed.

2 Reinforcement Learning from Expert Demonstrations

RLED considers two possible knowledge sources to guide the RL process, 1) prior knowledge, where demonstrations are provided before the RL process; 2) online knowledge, where demonstrations are occasionally provided during the RL process.

This type of knowledge complements the agent knowledge progressively acquired by interacting with the environment. RLED needs three general assumptions:

1. The agent has access to interactions with the environment.
2. Environment reward feedback is available.
3. At least one source of knowledge from demonstrations is available.

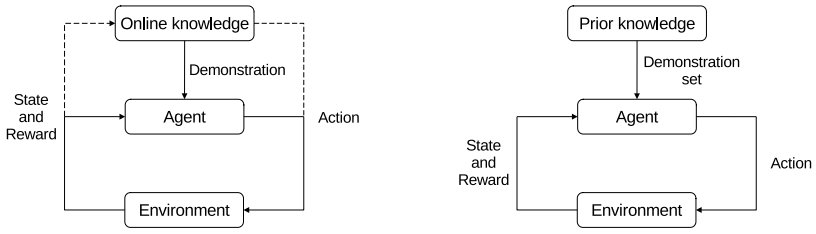

Fig. 1 Different sources of knowledge in RLED. These diagrams illustrate how different knowledge sources interact with the agent and environment to guide the RL process. Dashed line: Occasional interaction. In RLED with prior knowledge, the teacher provides a set of demonstration trajectories (before the RL process) to be used by the agent as a source of knowledge in the RL process. In RLED with online knowledge, the teacher occasionally provides a demonstration trajectory (by taking control over the agent) to be used by the agent as a source of knowledge in the RL process.

Similar to RL, RLED is formalized in the context of a Markov Decision Process (MDP) [85], which is defined as a tuple $M = \langle S, A, R, T, \gamma \rangle$, where S is the set of states, A is the set of admissible control actions, a reward function $R(s, a)$, a probabilistic transition function $T(s'|s, a)$ for the stochastic case or a deterministic transition function $T(s, a) = s'$ for the deterministic case, and a discount rate $\gamma \in [0, 1)$. In each state $s \in S$, the agent takes a control action $a \in A$, then a reward R is received, and a next state s' is reached in the environment, which is determined by T. The objective is to find a control policy $\pi(a|s)$ for the stochastic case or a control policy $\pi(s) = a$ for the deterministic case, that maximizes the discounted cumulative reward defined by

$$G = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

then, the optimization problem to be solved can be written as

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}[G|\pi].$$

Knowledge from demonstrations is leveraged by an expert demonstrator policy π_d (not necessarily optimal) to guide the learning process. When following π_d, for a demonstrated state $s^d \in S$ is demonstrated a control action $a^d \in A$. Depending on the algorithm, the demonstration set contains at least one trajectory, which is conformed of a temporal sequence of demonstrated states $\zeta^d = \{s_t^d\}$, can also be conformed of a temporal sequence of demonstrated state-action pairs, $\zeta^d = \{(s_t^d, a_t^d)\}$, a temporal sequence of demonstrated state-action-reward tuple $\zeta^d = \{(s_t^d, a_t^d, r_t^d)\}$, or a temporal sequence of demonstrated state-action-reward-next state tuple $\zeta^d = \{(s_t^d, a_t^d, r_t^d, s_{t+1}^d)\}$, where t index time.

We can assume that the teacher is an expert subject as he needs to have relevant knowledge about the task or at least some experience in resolving the task. It is important to emphasize that although we can consider the teacher an expert subject, it does not mean that the provided demonstrations trajectories follow an optimal control policy. Instead, in many cases, mainly where humans provide the demonstration is more natural to assume that the demonstrations follow a sub-optimal control policy. This is due to natural human reasons like occasional mistakes, small tremor movements, and coordination issues.

The expected value in the state s following the control policy π is known as state-value function and is defined as

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G|s] \tag{1}$$

in the same way, the expected value of taking a control action a in the state s following the control policy π is known as the action-value function and is defined as

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G|s, a] \tag{2}$$

then, the optimal values are given by $V^*(s) = \max_{\pi} V^{\pi}(s)$ and $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$. The advantage is defined as

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s).$$

A fundamental property of the equations (1) and (2) is that they satisfy particular recursive relations [85] that allow expressing them as Bellman equations [5] by following

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R(s_t, a_t) + \gamma V^{\pi}(s_{t+1})\right]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1})\right].$$

There are three base approaches to solve the RL part of the RLED problems, value-based methods, policy-based methods, and actor-critic methods. We briefly mention these three approaches.

- Value-based methods estimate a value-function. Their simplicity and ease of implementation distinguish these methods, so they are, in general, the most popular approach. RLED value-based methods extend the most popular methods, such as Q-Learning [97], SARSA[72], Deep Q-Networks (DQN) [61], Double DQN (DDQN) [90], Prioritized Dueling Double Deep Q-Networks (PDD DQN) [75], and Dueling Network Architectures for Deep Reinforcement Learning [94].
- Policy-based methods directly estimate the control policy. These methods are useful with continuous or very large action spaces and allow learning stochastic control policies. RLED policy-based take as base popular methods, which include, Natural Policy Optimization (NPG) [38], Trust Region Policy Optimization (TRPO) [76] and Proximal Policy Optimization (PPO) [77].
- Actor-critic methods are a combination of value-based and policy-based methods, where "actor" refers to learning the control policy, while "critic" refers to learning the value function. Some RLED actor-critic use as base method one of the most popular methods today, Deep DPG (DDPG) [54], which combines the ideas of DPG [80] and DQN. Another method used as a base in RLED actor-critic is Asynchronous advantage actor-critic (A3C) [60].

We classify the different RLED methods based on how the algorithms use the demonstration trajectories for learning the control policy. The discussed RLED methods are listed in Table 1.

Classification			Algorithm	Base algorithm
Prior Knowledge	Biased Exploration		HAT [88]	SARSA [72] and Q-Learning [97]
			CHAT [95]	SARSA [72] and Q-Learning [97]
			DRoP [96]	Q-Learning [97]
			Shaping [10]	Q-Learning [97]
			Introspective Shaping [52]	Q-Learning [97]
	Extended Optimization Criterion	With Pre-training	DQfD [33]	DQN [61]
			DDPGfD [91]	DDPG [54]
			CoL [23]	DDPG [54]
			DAPG [68]	NPG [38]
			Pre-trained Model [15]	DQN [61] and A3C [60]
			NAC [19]	Soft Q-Learning [30]
		Without Pre-training	DPID [11]	DPI [49]
			Accelerated DQN [47]	DQN [61]
			Overcoming Exploration [62]	DDPG [54]
			POfD [39]	TRPO [76] and PPO [77]
			Soft Expert Guidance [36]	PPO [77]
		Distributed	Ape-X DQfD [66]	DQN [61]
			EA-ACKTR [21]	ACKTR [99]
			R2D3 [27]	DQN [61]
			FS-ER and DFS-ER [101][100]	DQN [61] and ACKTR [99]
	Episode Initialization		Backplay [73][70]	PPO [77] and A3C [60]
	Reasoning		RfD [89]	Q-Learning [97]
Online Knowledge	Query for Demonstrations		EfD [82]	Q-Learning [97]
			ARLD [12]	DQN [61]
			Minimal Human Effort [71]	DDPG [54]

Table 1 RLED methods classified by how demonstrations are used.

2.1 RLED with prior knowledge

In RLED methods from prior knowledge, the teacher provides a set of demonstration trajectories previous to the reinforcement learning process. These demonstration trajectories are stored to help the agent at the learning process to acquire a similar teacher's behavior.

There are different ways to use prior knowledge. We classified the most relevant methods by how prior demonstrations are used in: biased exploration, extended optimization criterion, episode initialization, and reasoning.

In the first classification, biased exploration, the main concern in these methods is how to obtain a control policy able to follow the best parts of the teacher's behavior. Therefore, the RL agent is encouraged to explicitly explore and evaluate the states and actions in the demonstration trajectories. This way, the agent learns when it is better to follow states and actions from the demonstration trajectories or find a better choice.

In the second classification, extended optimization criterion, the optimization problem is complemented with different terms to encourage the agent to have a similar teacher's behavior with better performance, leading to an implicit exploration and evaluation of the demonstrated states and actions. The immediate idea to extend the optimization criterion could be to add a pure IL loss to force the agent to follow the teacher's behavior. Nevertheless, this kind of technique based on a pure IL loss can potentially over-fit and lack generalization, as mentioned by Lakshminarayanan, Ozair, and Bengio in [47]. Furthermore, the expert's knowledge must be persistent throughout the control policy's learning and not just be a starting point. We subdivide this second classification into three categories: with pre-training phase, without pre-training phase, and distributed. Methods with the pre-training phase divide the learning process into two phases: the pre-training and the fine-tuning phases. In the pre-training phase, only demonstration trajectories are used for learning, while in the fine-tuning phase, the demonstration trajectories and the interactions with the environment are used. Methods without the pre-training phase use the demonstration trajectories along the learning process to keep the agent's control policy close to the teacher's behavior. Methods in the category of distributed give the possibility to use distributed RL. Some of these methods are an extension of the category with pre-training phase and without pre-training phase. Another thing to consider in the extended optimization criterion classification is the importance of the trade-off between RL and the additional terms. The correct balance between RL random exploration and explore demonstrations leads to better results searching for an optimal control policy, even when sub-optimal demonstrations are provided [53].

Episode initialization is the third classification, where the main idea is to use the demonstration trajectories as a starting point in the learning process episodes. The episode is initialized from a specific demonstrated state near a state with high reward and consecutively initialized in further episodes from a more distant demonstrated state. This way, the typical RL process runs as

usual with high possibilities to reach the high reward state without explicit bias of exploration or an extended optimization criterion.

The fourth classification, reasoning, is about a completely different idea, mainly related to how humans approach tasks by causal reasoning. In this method, the task is decomposed to identify relationships between cause and its effects.

2.1.1 Biased exploration

Taylor, Suay, and Chernova in [88] propose Human-Agent Transfer (HAT), a three-step method.

1. The agent records the demonstrations of all the state and action transitions.
2. From the recordings and through a decision tree learning method, a summarized control policy is obtained in the form of a list of rules.
3. Rules are transferred to an RL agent to improve further and outperform the summarized control policy.

Three methods are proposed to transfer the summarized control policy to an RL agent, Value Bonus, Extra Action (both initially proposed in [87]), and Probabilistic Policy Reuse. Value Bonus assigns a Q-value to the summarized policy's actions, forcing the RL agent to execute these actions for a number of episodes. Extra Action gives the RL agent the option to decide between taking a pseudo-action (the summarized control policy actions) or taking a random action. The RL agent learns through exploration when to follow the summarized control policy and when to take a different control action. Probabilistic Policy Reuse is similar to ϵ-greedy [83] as it assigns a probability ψ of taking the actions of the summarized control policy, a probability ϵ of taking a random control action, and a probability $1-\psi-\epsilon$ of taking the greedy action.

Confidence-HAT (CHAT) by Wang and Taylor [95] extends HAT, modifying its second and third steps. In the second step, three confidence-aware classifiers are proposed to be trained from the recorded demotions, Gaussian Process HAT (GPHAT)

$$C_i(x) = \exp\left(-((x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)) - (\ln 2\pi|\Sigma_i| - 2\ln P(\omega_i)\right),$$

where x is a state vector, Σ_i is the covariance matrix of class i, μ_i is the mean of data of class i, and $P(\omega_i)$ is a typical Gaussian model with ω_i as the predicted label; Neural Network HAT (NNHAT)

$$C_i(x) = \frac{1}{\sum_i \exp(\theta_i^\top x)} \begin{bmatrix} \exp(\theta_1^\top x) \\ \exp(\theta_2^\top x) \\ \vdots \\ \exp(\theta_i^\top x) \end{bmatrix},$$

where θ_i is the neural network weight vector corresponding to the i-th output of a "softmax" layer; and Decision Tree HAT (DTHAT), which uses each

leaf node's accuracy as an estimate of confidence. In the third step, Probabilistic Policy Reuse is used to transfer the summarized control policy to an RL agent, and a restriction is added to execute the actions suggested by the demonstrations when the confidence is above a confidence threshold.

Dynamic Reuse of Prior (DRoP) [96] is another extension of HAT. DRoP adds an online confidence measure using a temporal difference model to analyze the performance in every source action for a given state, with the update rule

$$C(s) \longleftarrow (1 - F(\eta)) \times C(s) + F(\eta) \times [H(R) + \gamma \times C(s')],$$

where η is the update parameter, $F(\eta)$ and $H(R)$ depend on the type of confidence to update, as there are two types, confidence prior knowledge $CP(s)$ and confidence Q knowledge $CQ(s)$. $CP(s)$ can be updated in two different ways, in one way $F(\eta) = \eta \times P$ and $H(R) = R$, and in the other way $F(\eta) = \eta$ and $H(R) = \frac{R}{R_{max}} \times P$. If $CQ(s)$ is being updated $F(\eta) = \eta$ and $H(R) = R$, where P is a neural network or a Gaussian model similar to CHAT, and R_{max} is the absolute maximum reward value. The source actions are then selected using a probability distribution to be greedy with respect to the confidence measure, by taking actions with higher confidence between the demonstration and the RL agent experience, and by interacting with the environment following hard decision

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

Alternatively, a soft decision to allow exploration of actions with lower confidence by following

$$AS = \begin{cases} P_1 = \frac{\tanh(RCQ(s))+1}{\tanh(RCP(s))+\tanh(RCQ(s))+2} \\ P_2 = \frac{\tanh(RCP(s))+1}{\tanh(RCP(s))+\tanh(RCQ(s))+2} \end{cases}$$

where P_1 is the Q knowledge, P_2 is the prior knowledge. CHAT and DRoP were extended to a multi-agent framework in [4].

Different from HAT and its extensions, Brys et al. in [10] proposed an approach through reward reshaping, which introduces a function to complement the base reward function, this defines a new reward function

$$R_F(s, a, s', a') = R(s, a) + F(s, a, s', a')$$

that stimulates the exploration of the RL agent towards the states and actions of the demonstrations. $F(s, a, s', a')$ is defined by

$$F(s, a, s', a') = \gamma\phi(s', a') - \phi(s, a),$$

where the potential function $\phi : S \to \mathbb{R}$ models the distribution of the states and actions of the demonstrations using a non-normalized multi-variate Gaussian defined by

$$\phi(s, a) = \max_{s^d, a^d} \exp\left(-\frac{1}{2}(s - s^d)^\top \Sigma^{-1}(s - s^d)\right).$$

The potential function $\phi(s,a)$ is high when an action a^d has been demonstrated for a state s^d near the current state s, and on the other hand, it is low when there is no action demonstrated near the current state s. This idea was also applied to IRL in [81]. Li, Brys, and Kudenko [52] extended the idea of reward reshaping through potential function for RLED by proposing an introspective RL agent. The introspective RL agent records its state-action decisions and experience during training in a priority queue and estimates a Monte Carlo q-value $\hat{q}(s^d,a^d)$ to calculate a potential function

$$\phi(s,a) = \rho \max_{s^d,a^d} \exp\left(-\frac{1}{2}(s-s^d)^\top \Sigma^{-1}(s-s^d)\right)\hat{q}(s^d,a^d),$$

where ρ is a scaling factor to control the strength of the biased exploration. If the RL agent finds a state-action with a higher action-value than the lowest action-value in the queue, the lowest is replaced by the higher. The introspective RL agent does not necessarily need optimal demonstrations, but using the prior knowledge from demonstrations gradually improves the training process. With a more general idea of prior knowledge from multiple experts, Gimelfarg, Sanner, and Lee [22] used a Bayesian framework to be applied in Reward Shaping compatible with [10].

2.1.2 Extended optimization criterion with pre-training phase

Perhaps one of the most outstanding works in RLED is presented by Hester et al. in [33], where Deep Q-learning from Demonstrations (DQfD) is proposed, which combines PDD DQN with IL. DQfD begins with a pre-training phase of the control policy, where the RL agent acts as a supervised learning algorithm. During the pre-training phase, only the data of the demonstrations stored in the replay buffer are available, and there is no interaction with the environment. The agent samples mini-batches from the replay buffer to update the network weights by following the optimization criterion

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q),$$

where λ is a weight parameter, $J_{DQ}(Q)$ is a 1-step double Q-learning loss function, $J_n(Q)$ is an n-step double Q-learning loss function, $J_{L2}(Q)$ is an L2 regularization loss function, and $J_E(Q)$ is a supervised large margin classification loss function defined by

$$J_E(Q) = \max_a [Q(s,a) + l(a^d,a)] - Q(s,a^d),$$

where $l(a^d,a)$ is a marginal function that is 0 when $a = a^d$ and positive otherwise. After the pre-training phase, the RL agent begins to interact with the environment, stores its experiences in the same replay buffer containing the demonstrations, and progressively updates the network's weights to outperform over unseen states and actions. Due to the supervised large margin classification loss, the RL agent tends to overfit the demonstration trajectory's

action-values, which are always higher than any other control action encountered by the agent, making it impossible for the RL agent to improve upon the demonstrations.

DDPG from Demonstrations (DDPGfD) [91] by Vecerik et al. extends DDPG to benefit from the demonstrations. DDPGfD, like DQfD, divides learning into two phases, the pre-training phase, where the agent stores the demonstration's trajectories in a replay buffer to learn in a supervised manner, and the fine-tuning phase, where the agent learns by interacting with the environment. DDPGfD extends DDPG to make use of prioritized experience replay to sample transitions across the demonstrations and the agent data generated by interacting with the environment. It uses a 1-step $J_1(\theta^Q)$ and an n-step $J_n(\theta^Q)$ return losses to update the critic function to help spread the sparse rewards, it learns multiple times per environment step, and it uses an L2 regularization loss on the actor $J^a_{L2}(\theta^\pi)$ and on the critic $J^c_{L2}(\theta^Q)$ to stabilize learning. The actor and critic loss functions are

$$J_{critic}(\theta^Q) = J_1(\theta^Q) + \lambda_1 J_n(\theta^Q) + \lambda_2 J^c_{L2}(\theta^Q)$$
$$J_{actor}(\theta^\pi) = -J_A(\theta^\pi) + \lambda_2 J^a_{L2}(\theta^\pi),$$

where $J_A(\theta^\pi)$ is the DDPG actor Q-loss. Unlike DQfD, which can only be applied in domains with discrete action spaces, the major contribution of DDPGfD is to be applicable in domains with continuous action spaces.

Similar to DDPGfD, Cycle-of-Learning (CoL) [23] proposed by Goecks et al. is based on DDPG. It combines a behavior cloning loss $J_{BC}(\theta^\pi)$, the DDPG actor Q-loss $J_A(\theta^\pi)$, a 1-step loss $J_1(\theta^Q)$, and an L2 loss for the actor $J^a_{L2}(\theta^\pi)$ and for the critic $J^c_{L2}(\theta^Q)$ in a single loss function to update the critic and the actor. It takes advantage of expert demonstrations stored in an expert's buffer. The behavior cloning loss is

$$J_{BC}(\theta^\pi) = \frac{1}{2}(\pi_\theta(s) - a^d)^2.$$

The complete loss function is

$$J(\theta^Q, \theta^\pi) = \lambda_1 J_{BC}(\theta^\pi) + \lambda_2 J_A(\theta^\pi) + \lambda_3 J_1(\theta^Q) + \lambda_4 J^c_{L2}(\theta^Q) + \lambda_5 J^a_{L2}(\theta^\pi).$$

Before interacting with the environment, both the actor and the critic networks are updated in the pre-training phase. Once the pre-training phase ends, the agent starts interacting with the environment and stores its experiences in a separate buffer; when the expert's buffer is full, the fine-tuning phase takes a fixed ratio of 25% expert's buffer samples and 75% agent's buffer samples. Col can be applied for continuous action spaces.

In [68], a method called Demonstration Augmented Policy Gradient (DAPG) is proposed, incorporating a pre-training phase based on IL and a fine-tuning phase based on policy gradient. In the pre-training phase, a parameterized policy π_θ imitates the demonstration's behavior, which corresponds to the following maximum-likelihood optimization problem

$$\max_\theta \sum_{(s^d, a^d) \in \zeta^d} \ln \pi_\theta(a^d | s^d).$$

This provides the RL agent with a good initialization and greatly reduces the sample complexity. The RL fine-tuning phase extends the NPG gradient to optimize π_θ by following

$$\nabla_\theta J = \sum_{(s,a)\in\zeta^\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a) + \sum_{(s^d,a^d)\in\zeta^d} \nabla_\theta \ln \pi_\theta(a^d|s^d) w(s,a),$$

where ζ^π is the trajectory obtained by following policy π, and $w(s,a)$ is the weighting function

$$w(s,a) = \lambda_0 \lambda_1^k \max_{(s',a'^\pi)} A^\pi(s',a')$$

where λ_0 and λ_1 are hyper-parameters, and k is the iteration counter.

Cruz, Du, and Taylor in [15] used a multiclass-classification deep neural network trained from human demonstrations in the pre-training phase; the neural network implicitly learns the underlying features by only using the cross-entropy loss function. The network uses raw images of the domain as input, and it has a single output for each valid action. The learned weights and biases from the classification model are used to initialize a DQN or an A3C agent.

Gao et al. propose Normalized Actor-Critic (NAC) [19], where, unlike previous methods in this category, it does not use a supervised loss function in the pre-training phase; therefore, it gives greater robustness to imperfect or noisy demonstrations. First, the action-value function is parameterized by θ, then the state-value and the policy are parameterized by the action-value

$$V_Q(s) = \alpha \log \sum_a \exp\left(\frac{Q_\theta(s,a)}{\alpha}\right)$$

$$\pi_Q(a|s) = \exp\left(\frac{Q_\theta(s,a) - V_Q(s)}{\alpha}\right).$$

NAC extends soft policy gradient formulation from Soft Q-Learning [30] to obtain a gradient of the action-value function that reduces the action-values of the actions not included in the demonstration trajectories. The gradients for the actor and the critic are

$$\nabla_\theta J_{PG}(\theta) = \mathbb{E}_{s,a\sim\pi_Q}\left[(\nabla_\theta Q_\theta(s,a) - \nabla_\theta V_Q(s))(Q_\theta(s,a) - \hat{Q}(s,a))\right]$$

$$\nabla_\theta J_V = \mathbb{E}_s\left[\nabla_\theta \frac{1}{2}(V_Q(s) - \hat{V}(s))^2,\right]$$

where $\hat{Q}(s,a)$ and $\hat{V}(s)$ are estimates calculated by

$$\hat{Q} = R(s,a) + \gamma V_Q(s')$$

$$\hat{V}(s) = \mathbb{E}_{a\sim\pi_Q}\left[R(s,a) + \gamma V_Q(s')\right] + \alpha H(\pi_Q(\cdot|s)).$$

H is an entropy term, and α is a parameter to control the entropy term's relative importance and the reward. The pre-training phase uses only the demonstration trajectories, and the fine-tuning phase uses only the interactions with the environment. By avoiding supervised loss, NAC avoids overfitting on the demonstration trajectories.

2.1.3 Extended optimization criterion without pre-training phase

Lakshminarayanan, Ozair, and Bengio in [47] propose a simple but practical method that incorporates expert demonstrations to DQN by adding a weighted IL loss function to the original DQN loss function

$$J = J_{DQN} + \beta J_{imitation},$$

where the imitation loss is

$$J_{imitation} = \mathbb{E}_{(s^d,a^d)\in B^d} \log \frac{\exp\left(\tau^{-1} Q_{DQN}(s^d, a^d)\right)}{\sum_{a\in A} \exp\left(\tau^{-1} Q_{DQN}(s, a)\right)}$$

and $B^d \in \zeta^d$ is a random mini-batch of size τ. This imitation loss function ensures a high action-value when an action has been demonstrated in a particular state and a low action-value for all other not demonstrated actions.

A more advanced method by Nair et al. [62] proposes a method based on DDPG to overcome exploration. The method extends the original DDPG loss function $J_A(\theta_\pi)$ with a standard behavior cloning loss and adds a filter so that the IL loss is only applied to the states where the critic determines that the demonstrated actions are better than the actor's actions. The behavior cloning loss is

$$J_{BC}(\theta_\pi) = \sum_{(s^d,a^d)\in B^d} ||\pi_\theta(s^d) - a^d||^2 \mathbf{1}_{Q(s^d,a^d)>Q(s^d,\pi(s^d))},$$

where B^d is a minibatch, and the complete loss function for the actor is

$$J(\theta_\pi) = \lambda_1 J_A(\theta_\pi) - \lambda_2 J_{BC}(\theta_\pi).$$

Unlike DQfD, this method finds better control actions than those proposed by the demonstrations; in this algorithm, a random reset to demonstration states is incorporated to start a learning episode from a demonstration state.

Direct Policy Iteration with Demonstrations (DPID) [11] based on Direct Policy Iteration (DPI) [49]. DPID consists of constructing a cost-sensitive training set B^{RL} from a given sample distribution ρ over the set of states S and integrating it into the set of demonstration trajectories ζ^d, where the demonstrated states s^d are drawn from a distribution μ over S. The loss functions are

$$\hat{J}_{DPI}(\hat{\rho}, \pi) = \frac{1}{|B^{RL}|} \sum_{(s,a)\in B^{RL}} \left[\max_a \hat{Q}(s, a) - \hat{Q}(s, \pi(s))\right]$$

$$J_e(\hat{\mu}, \pi) = \frac{1}{|\zeta^d|} \sum_{(s^d,a^d)\in\zeta^d} \mathbb{I}(a^d \neq \pi(s^d)),$$

where $\hat{Q}(s_j, a) = \frac{1}{M}\sum_{j=1}^{M} R_j(s_j, a)$ is an action-value estimate using M rollouts. The complete loss function is

$$J(\hat{\rho}, \hat{\mu}, \pi) = \hat{J}_{DPI}(\hat{\rho}, \pi) + \lambda J_e(\hat{\mu}, \pi).$$

The loss classification function is minimized with respect to π. DPID works for specific policies with small state and action spaces.

Policy Optimization from Demonstration (POfD) by Kang, Jie, and Feng [39] extends policy gradient methods to guide RL agent's exploration near the demonstrated policy. POfD introduces a practical algorithm based on Jensen-Shannon Divergence that resembles a Generative Adversarial Network (GAN) [24]. The POfD algorithm minimizes the difference measure between the parameterized policy π_θ and the expert policy π_d. The new criterion to optimize is then redefined based on occupancy measures to exploit the demonstrations better and facilitate the new criterion's optimization. This loss function is formulated as

$$J_M(\theta_\pi) = \mathbb{D}_{JS}\left[\rho_{\pi_\theta}(s,a)||\rho_{\pi_d}(s,a)\right]$$

by optimizing the lower bound of the Jensen-Shannon Divergence, the above loss function is

$$J_M(\theta_\pi) = \sup_{D\in(0,1)} \mathbb{E}_{\pi_\theta}\left[\log(D(s,a))\right] + \mathbb{E}_{\pi_d}\left[\log(1-D(s,a))\right],$$

where $D(s,a) = \frac{1}{1+\exp\left(-U(s,a)\right)}$, and $U(s,a)$ is an arbitrary function. The complete loss function is

$$J(\theta_\pi) = -J_{VA}(\theta_\pi) - \lambda_1 H_{\pi_\theta}(\theta_\pi) + J_M(\theta_\pi),$$

where $J_{VA}(\theta_\pi) = \mathbb{E}_{\pi_\theta}[G|\pi_\theta]$ is the vanilla objective, and H_{π_θ} is a causal entropy term for regularization. POfD is a general method compatible with conventional policy gradient methods such as TRPO, PPO, and NPG.

Jing et al. propose Soft expert guidance from demonstrations [36] is based on a control policy constrained optimization. The optimization criterion is subject to a bounded tolerance factor such that the control policy π_{θ_k} always stays within an area closer to the demonstration policy π_d. The constraint limits the exploration region near the demonstration trajectories (not necessarily perfect) under a threshold. The optimization problem is

$$\begin{gathered}\theta_{k+1} = \arg\max_\theta \mathbb{E}_{\pi_{\theta_k}}[G|\pi_{\theta_k}] \\ s.t. \mathbb{D}\left[\rho_{\pi_{\theta_k}}(s,a)||\rho_{\pi_d}(s,a)\right] \leq \alpha_k \\ \mathbb{D}\left[\pi_{\theta_k}(a|s)||\pi_{\theta_{k+1}}(a|s)\right] \leq \delta\end{gathered}$$

where $\mathbb{D}$ is the divergence, $\rho_\pi(s,a) = \pi(a|s)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)$ is an occupancy measure, α_k and δ are the tolerance factors that control the discrepancy constraints. The above optimization problem is then approximately solved by linearizing around π_{θ_k} at each optimization step. The RL agent only updates the control policy when the constraint is not satisfied.

2.1.4 Distributed setup

Ape-X DQfD [66] combines DQfD with Ape-X DQN [34] to run on a distributed setup. Ape-X DQfD removes the pre-training phase and proposes a new transformed Bellman operator to define the temporal difference loss function

$$\begin{aligned} &JTD(\theta_k^Q, (p_i)_i = 1^N, \theta_k - 1^Q) \\ &= \sum_{i=1}^{N} p_i L\left(Q_{\theta_k}(s_i, a_i) - h\left(R(s_i, a_i) + \gamma \max_{a' \in A} h^{-1}(Q_{\theta_{k-1}}(s_i', a'))\right)\right) \end{aligned}$$

where $h(z) = sign(z)(\sqrt{|z|+1} - 1) + \epsilon z$ with $\epsilon = 10^{-2}$, $L(\cdot)$ is the Huber loss, which is typically applied in DQN and $p_1, ..., p_N$ are normalized priorities. To stabilize learning is used a temporary consistency loss function

$$J_{TC}(\theta_k^Q, (p_i)_{i=1}^N, \theta_{k-1}^Q) = \sum_{i=1}^{N} p_i L(Q_{\theta_k}(s_i', a_i') - Q_{\theta_{k-1}}(s_i', a_i')),$$

and a max-margin loss function for IL

$$J_{IM}(\theta_k^Q, (p_i)_{i=1}^N, \theta_{k-1}^Q) = \sum_{i=1}^{N} p_i e_i \left(\max_a \left[Q_{\theta_k}(s_i, a) + \lambda \delta_{a \neq a^d}\right] - Q_{\theta_k}(s_i, a^d)\right),$$

where $\lambda \in \mathbb{R}$ is the margin. $\delta_{a \neq a^d}$ is equal to one if $a \neq a^d$ and zero if not, e_i is part of the sampled trajectories, and it is equal to one if the transition is part of the best expert episode and it is equal to zero if not. The complete loss function is

$$J = J_{TD}(\theta_k^Q, (p_i)_{i=1}^N, \theta_{k-1}^Q) + J_{TC}(\theta_k^Q, (p_i)_{i=1}^N, \theta_{k-1}^Q) + J_{IM}(\theta_k^Q, (p_i)_{i=1}^N, \theta_{k-1}^Q).$$

Ape-X DQfD uses a fixed ratio of 75% of agent transitions and 25% of expert transitions to learn.

Expert Augmented ACKTR (EA-ACKTR) [21] is another distributed method, which is based on the Actor-Critic using Kronecker-Factored Trust Region (ACKTR) algorithm [99]. EA-ACKTR adds a new term to the original ACKTR loss function J_{A2C} to consider expert demonstration trajectories stored in an expert replay buffer, separated from the different actors' interactions with the environment. The new loss function is

$$J_{EA}(\theta_\pi) = \frac{\lambda_e}{|B^d|} \sum_{(s^d, a^d) \in B^d} A(s^d, a^d) \log \pi_\theta(a^d | s^d),$$

where B^d is a mini-batch of demonstrated state-action pairs. The complete loss is

$$J(\theta_\pi) = J_{A2C}(\theta_\pi) + J_{EA}(\theta_\pi).$$

Gulcehre et al. combined Recurrent Replay Distributed DQN (R2D2) [40] with demonstrations in [27], their R2D2 from Demonstrations (R2D3) method.

R2D3 has several actors running independent copies on the behavior; the actor's interactions with the environment are stored into a shared actor buffer, where they are globally prioritized using a mixture of max and mean of the TD-error. The demonstrations are stored in a separate buffer and also prioritized. The buffer priorities are updated with a learner process sampling batches from both buffers simultaneously and applying a fixed stochastic ratio to control the proportion of demonstrations and actor experiences from the environment. Also, the actors periodically request the latest network weights from the learner to update their behavior.

Yeo, Oh, and Lee sped up the DQfD and EA-ACKTR training process using dual replay buffer management and frame skipping, with their Dynamic Frame Skipping-Experience Replay (DFS-ER) and Frame Skipping-Experience Replay (FS-ER) [101][100]. FS-ER and DFS-ER use dual replay buffers, where the demonstration trajectories are stored in a human replay buffer, and the agent interactions with the environment are stored in an actor buffer; each buffer has an independent sampling policy. FS-ER uses an online frame skipping scheme to take advantage of the hole demonstration trajectory, while DFS-ER uses a dynamic frame skipping scheme to manage data from repeated actions on the demonstration trajectories.

2.1.5 Episode initialization

Salisman and Chen in [73] and in parallel Resnick et al. in [70] propose to use the states of a single demonstration as a starting point for each episode in the RL process. Learning starts with an episode initialization from the state of the demonstration trajectory closest to the positive reward. Then, each next learning episode is initialized from a state of the demonstration trajectory more distant from the positive reward. This backward initialization makes it easy to solve a task with a very sparse reward since the agent in each learning episode faces an easy exploration problem with a very high probability of finding a positive reward.

The discussed method by Nair et al. [62] can also be considered as an episode initialization method, as it incorporates a random reset to demonstration states that allows starting a learning episode from a random state of a demonstration trajectory.

2.1.6 Reasoning

Torrey introduces a new idea to RLED called Reasoning from Demonstration (RfD) [89]. RfD is based on how humans reason and acquire knowledge from task demonstrations. Given a demonstration, the RfD agent decomposes the task as follows:

– Object and events: the set of objects present in a particular state s, each object with an observed position, velocity, and a region the object occupies in the environment. The event is the set of object interactions observed in the state transition from s to s'.

- Environment: it consists of a state space, action space, and an unknown probability distribution over state transitions.
- Task: it is defined over the environment with four subsets of state spaces to choose from: the begin, the end, the success feedback, and the failure feedback.
- Objectives: these are desirable object interactions that contribute to task success.
- Anti-objectives: these are undesirable object interactions to be avoided as they contribute to task failure.

RfD develops a theory, a map, and a set of policies during the training process. The theory is a set of cause-effect hypotheses with simple assumptions, where causes implicate effects, object interactions are potential causes, and environment feedback of success or failure are potential effects. The map is a graph of environment regions and how they are connected. The policy then evaluates the actions with respect to a possible event. RfD can deal with low-quality imperfect demonstrations.

2.2 RLED with online Knowledge

In RLED methods with online knowledge, the agent interacts with the teacher by querying what behavior to follow upon observing states. When querying, the teacher takes control over the agent for a few consecutive steps. The online demonstrated trajectories are stored as other RLED methods and used by the agent to improve its performance. After the querying period, the agents take control back, and the RL process continues until the next query. The main concern of RLED from online knowledge is when to ask for a demonstration, as the teacher's time can be considered more valuable than the agent's time.

2.2.1 Query for demonstration

Subramanian, Isbell Jr., and Thomaz proposed Exploration from Demonstration (EfD) in [82], which guides the agent's exploration through the more relevant statistical measures of the learning algorithm to help it identify influential regions. It is based on Q-Learning with function approximation, where $Q_\theta(s,a) = \phi(s,a)^T\theta$. Influence is computed as a combination of Leverage and Discrepancy measures. Leverage is a measure of how far a specific observation is from the convex hull of known observations. Leverage is calculated with

$$H = X(X^TX)^{-1}X^T,$$

where X is a matrix of independent variables of size $n \times k$, with n number of observations and k number of features, X is populated with the state-action features $\phi(s,a)$. The diagonal elements h_{ii} of the matrix H represent the leverages, which describe each dependent variable value's influence on the fitted value from observation i. The discrepancy is a measure of how much

an observation contributes towards model error. For the observation i, the discrepancy E_i is calculated as

$$E_i = \frac{\delta_i}{\sqrt{MSE_{(i)}(1-h_{ii})}},$$

where $MSE_{(i)}$ is the mean squared error for the model based on all observations excluding sample i, and it is calculated with the mean squared error MSE

$$MSE_{(i)} = \frac{(n-p)MSE - \frac{e_i^2}{(1-h_{ii})}}{n-p-1},$$

where MSE is the mean squared error, n is the number of samples, p the number of independent variables, and δ_i is the TD error for sample i. For every MDP transition, the Leverage and Discrepancy are calculated and compared against a threshold. If the influence exceeds the threshold, an influential state is identified as influential, and a demonstration is queried as it needs more exploration in influential states.

Active Reinforcement Learning with Demonstrations (ARLD) is a framework proposed by Chen et al. in [12] and based on DQN. ARLD first estimates the uncertainty in current states, and then it compares the uncertainty in the current state. . If the current state uncertainty exceeds the current state's estimation, the agent queries the teacher for a demonstration, which will be used along with a supervised loss and the usual DQN loss to improve the agent's performance. They proposed two query methods based on Q-values uncertainty estimation: diverge and variance. The first proposed method to estimate the uncertainty is the divergence of bootstrapped DQN, which is based on bootstrapped DQN [64], where a value function head $Q_\theta^k(s,a)$ is built from $k \in \mathbb{N}$ bootstrapped estimates of the Q-value and trained against its target network $Q_{\theta^-}^k(s,a)$. To measure the divergence, they use the Jensen-Shannon divergence between the bootstrapped heads and use softmax normalized Q-values to obtain a policy distribution for each value function head $Q_\theta^k(s,a)$,

$$\pi_\theta^k(a|s) = \frac{\exp Q_\theta^k(s,a)}{\sum_{a'} \exp Q_\theta^k(s,a')}.$$

To estimate the Jensen-Shannon divergence between policy distributions of each value function head

$$U_D = \mathbb{D}_{JS}(\pi_\theta^1, \pi_\theta^2, ..., \pi_\theta^k).$$

The second proposed method to estimate the uncertainty, the predictive variance of noisy DQN, is based on noisy networks [18] [65]. Noisy networks are neural networks whose weights and biases are perturbed by a parametric function of noise. In this case, noisy networks are used as an exploratory policy to estimate the uncertainty using its predictive variance. The noisy network output layer is

$$Q_\theta(s,a) = w_a \phi(s) + b_a,$$

where $\phi(s)$ is the input to the last layer, w_a and b_a are the variables corresponding to the action a, where $w_a \sim N(\mu^{w_a}, \text{diag}((\sigma^{w_a})^2))$ and $b_a \sim N(\mu^{b_a}, (\sigma^{b_a})^2)$ with μ^{w_a}, σ^{w_a}, μ^{b_a} and σ^{b_a} the mean and noise level parameters of w_a and b_a respectively. By deriving the variance of the last equation, the predictive variance is

$$\text{Var}\left[Q(s,a)\right] = \phi(s)^T \text{diag}((\sigma^{w_a})^2)\phi(s) + (\sigma^{b_a})^2.$$

The uncertainty can be calculated with the variance of the action with the largest Q-value:

$$U_V = \text{Var}\left[Q\left(s, \arg\max_a Q(s,a)\right)\right].$$

When the agent finds states with high uncertainty, the RL process stops, and the agent query for a demonstration trajectory. This way, the agent avoids catastrophic actions and provides the data as demonstration trajectories to accelerate the RL process.

Rigter, Lacerda, and Hawes proposed an algorithm of minimal human effort by switching control between a human and the agent in [71]. The control is handled by the human when the agent performance is poor, and then the human-in-the-loop provides a demonstration trajectory. The agent must decide whether to ask for a demonstration or to use an already learned control policy, as they assume a cost for the human providing a demonstration. The selection of who handles control is formulated as a contextual Multi-Armed Bandit (MAB), where a continuous correlated beta process estimates MAB probabilities. For the k^{th} episode, a control policy $\pi_k \in \mathit{\Pi}$ is chosen, where $\mathit{\Pi} = \{\pi_h, \pi_{a,i}, ..., \pi_{a,n}\}$. The control policy may be a human following policy π_h or a control policy $\pi_{a,i}$ learned by using DDPG from the online demonstrations and interactions with the environment. The selection of the control policy follows

$$\pi_k = \arg\min_{\pi_i \in \mathit{\Pi}} \hat{R}_h(s_{k,0}, \pi_i),$$

where $s_{k,0}$ is the initial state for the k^{th} episode, and the cost function $\hat{R}_h(s_{k,0}, \pi_i)$ for using the control policy π_i is defined as:

$$\hat{R}_h(s_{k,0}, \pi_i) = \begin{cases} (1 - \hat{p}(s_{k,0}, \pi_i) - \alpha\hat{\sigma}(s_{k,0}, \pi_i)))R_f + R_d \longrightarrow \pi_i = \pi_h \\ (1 - \hat{p}(s_{k,0}, \pi_i) - \alpha\hat{\sigma}(s_{k,0}, \pi_i)))R_f \longrightarrow \pi_i \neq \pi_h \end{cases},$$

where R_f and R_d are the cost of human time required to recover from a failure state and the cost of human time invested per online demonstration trajectory provided, $\hat{p}(s_{k,0}, \pi_i)$ and $\hat{\sigma}(s_{k,0}, \pi_i)$ are the probability estimate and the standard deviation estimate of success for the control policy at the k^{th} episode, and α is the contextual MAB exploration and exploitation factor. The proposed reward is analogous to an upper confidence bound algorithm, which helps to determine when the teacher should take control upon the agent, this way this method minimizes the cost of bothering the human teacher by estimating the performance of each control policy and then only asking for demonstrations when it is needed.

3 Applications of RLED

Video games are typically used to evaluate RL algorithms. Nevertheless, some video games represent a real challenge because of their complexity. An effort to solve this challenging video game has been made through demonstrations, like the Atari Grand Challenge dataset [46], a large and diverse dataset of human Atari 2600 demonstrations. The dataset consists of five challenging games with almost 45 hours of gameplay: Video Pinball, Q*bert, Space Invaders, Ms. Pacman, and Montezuma's Revenge. Similar to the Atari Grand Challenge dataset, the Minecraft demonstrations dataset (MineRL) [29] is a large-scale, simulator paired dataset of human demonstrations with 60 million automatically annotated state-action pairs and over 500 hours of recorded human demonstrations to be used along with the Minecraft simulator Malmo [37]. MineRL includes six tasks with various research challenges, including open-world multi-agent interactions, long-term planning, vision, control, navigation, and subtask hierarchies. Recently the MineRL dataset and Malmo have encouraged the development of RLED algorithms in the Machine Learning community by running the MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors [28][59].

Two challenging Atari games, Montezuma's Revenge and Pitfall, were successfully solved with superhuman performance and exceeded human expert performance. The two video games were solved using the episode initialization Backplay method as part of their proposed algorithm Go-explore [17]. Other RLED methods like Ape-X DQfD and EA-ACKTR also tried to solve Montezuma's Revenge with poor performance, as they only cleared the first stage.

A classical application of RL, robotics, is an area where RLED has shown to help in real-world tasks. In [91], DDPGfD is successfully applied in four simulated robot insertion tasks: classic peg-in-hole, hard-drive insertion into a computer chassis, two-pronged deformable plastic clip insertion into a housing, and a cable insertion. The robotic hand is a Sawyer with 7 degrees of freedom. One of the simulated tasks, the hard-drive insertion, is tested using a physical version of the same robot. For the physical insertion task, demonstration trajectories were provided by using kinesthetic teaching. For the simulated insertion tasks, demonstration trajectories were provided by an agent running a hard-coded joint space P-controller to match the simulated robot's joint positions with the actual robot's joint position.

Similarly, DQfD is adapted to solve a robotic precision insertion problem in [98]. Insertion is divided into two phases; pose alignment and peg-in-hole insertion. The robot system consists of a 3 degree of freedom manipulator, a 4 degree of freedom adjustable platform, 3 microscopic cameras, and a high precision force sensor. Demonstrations are collected by a human doing kinesthetic teaching.

Two different dexterous multi-fingered robotic hands are used to show the potential of DAPG in [103]. A ROBEL 3-fingered manipulator hand [1] with 9 degrees of freedom and a 4-fingered Allegro hand with 16 degrees of freedom.

Both robotic hands performed three different tasks on rigid and deformable objects: valve rotation, box flipping, and door opening. With a small number of kinesthetic demonstrations provided by a human, the RL process is dramatically accelerated.

Ophthalmic microsurgery is a commonly performed surgical procedure, where deep anterior lamellar keratoplasty (DALK) is a novel procedure of partial thickness corneal transplantation. DALK is particularly promising but challenging to perform due to this technique requires the microsurgeon to have very high precision. The surgeon inserts a needle to separate with air the stroma and epithelium from the Descemet's membrane and endothelium. In [42], an RLED approach based on DDPGfD is used for *ex vivo* autonomous robotic DALK needle insertion, where experienced corneal surgeons provide demonstration trajectories.

In Natural Language Processing (NLP), a task-oriented Conversational AI based on DQfD is presented by [26], aiming to simulate a dialog to help users achieve booking flights or hotels. The environment is built upon a novel dialog system framework with large state and action spaces. The booking task's nature is a sparse reward, as the agent only receives a positive or negative reward (depending on the user goal) at the end of the dialog. These settings are compatible with an RLED approach, and therefore, DQfD is used to successfully improve the dialog policy, commonly a hand-written rule-based policy. The same Conversational AI is later improved in [25] to work under weak and cheap expert demonstrations.

In Finance, a dynamic pricing framework is proposed by [55] for e-commerce platforms. The dynamic pricing problem is solved with RLED using the real E-commerce platform; therefore, the RL process should not start with a poor performance to avoid capital loss. DQfD and DDPGfD are applied for discrete and continuous pricing to optimize the long-term revenue and avoid capital loss. Demonstration trajectories are provided by previous control policies evaluating historical sales data. The RLED strategy successfully outperforms the common manual markdown pricing strategy.

In financial security investment, the quantitative trading strategies are based on automated tools using mathematical models. In [56], they propose an adaptive trading framework for the quantitative trading problem. The agent uses the ideas of two RLED methods, the pre-training strategy from DDPGfD and the behavior cloning loss from Overcoming Exploration. The Recurrent Deterministic Policy Gradient [32] is used as the base algorithm. A typical quantitative trading strategy is used to provide the Demonstration trajectories. This method outperforms the most widely used methods in quantitative trading.

In Urbanism, a traffic signal control is designed by adapting the DQfD losses into the Advantage Actor-Critic (A2C) algorithm. An effective controller [14] that adjusts timing plans according to traffic dynamics provides the demonstration trajectories. With this extended version of A2C from demonstrations, the learned policy can outperform the teacher control policy.

4 Challenges

We would like to emphasize the addressed challenges by RLED in real-world implementations while mentioning the remaining challenges for this approach that can be addressed in the future through more advanced forms of guidance methods (not necessarily human-guided). These challenges of real-world RL applications are presented in almost all systems to a certain degree. Different works survey the challenges for real-world applications in RL [43][45][3] [16][104], while many others tackle these challenges separately.

4.1 Addressed RLED's challenges

The discussed RLED methods present a feasible way to implement RL in real-world systems by addressing essential RL challenges.

4.1.1 Sample efficiency

Sample efficiency is the amount of data needed to learn a good control policy. Standard DRL methods require lots of data from exploration, which is not sample-efficient. On the other hand, IL methods with data from demonstrations require a considerably smaller amount of data than RL, which is, more sample-efficient. RLED methods improve RL sample efficiency using data from demonstrations. The provided demonstration trajectories should cover a significant amount of relevant states, state-actions pairs, state-action-reward tuples, or state-action-reward-next state tuples (depending on the algorithm) to have a considerable impact on the RL process. A small set containing relevant demonstration trajectories is enough to accelerate the learning process in most cases.

4.1.2 Dimensionality

When the dimension of a particular space is considerably large, it is commonly known as "the curse of dimensionality" [6]. Powell [67] identifies three different spaces affected by the curse of dimensionality, the state space, the action space, and the outcome space. Common RL agents require large amounts of data acquired by exploration to learn a good control policy with high dimensionality. Dimensionality is then directly connected with the sample efficiency. RLED methods take advantage of the ability of standard RL methods to learn in high-dimensional spaces.

4.1.3 Safety

Safety refers to the process of learning a policy that maximizes the expected reward and ensures consistent system performance while safety constraints

are respected [20] to avoid risk states that can be dangerous for the system, humans, or surroundings.

Providing knowledge as demonstrations guides the RL process in the desired behavior, while excessive exploration is avoided. This excessive exploration is usual in standard RL methods, leading to exploring dangerous states and actions. Although RLED methods need some exploration, different strategies are used by the discussed methods in section 2 to improve the RL safety exploration indirectly. These strategies are: implicit and explicit biased exploration, which is prevalent in RLED methods with prior knowledge, interrupting the RL process to query for demonstrations, which is prevalent in RLED methods with online knowledge.

4.1.4 Reward specification

Sutton and Barto stated in [83] that RL applications' success strongly depends on how well the reward signal frames the application's designer's goal and how well the signal assesses progress in reaching that goal. In practice, it can be a complicated task to define a useful reward function. The most common is establishing a dense reward function using hand-coded functions, such as the Cartesian distance, which helps guide the RL agent to a control policy with smooth behavior, but this can potentially lead to unwanted behavior. A more natural approach is to express the reward as a sparse reward function, where a positive reward is assigned once the agent reaches a target state, leaving many states with reward zero. The use of demonstrations in RLED facilitates implementing a sparse reward specification by learning relevant states and actions that can successfully lead towards a target state.

4.2 Remaining RLED's challenges

Altough RLED presents a promising approach to behavioral learning, there is still room for theoretical and practical improvements. Relevant remaining challenges for the RLED approach are closely related to the robustness of the agent when facing different real-world situations. We consider it essential, and of the tremendous impact, the development of RLED methods can deal with or even take advantage of these situations to learn sufficiently robust control policies. The remaining challenges we consider are related to the environment and the structure of demonstration trajectories.

4.2.1 System delays

System delays are generally detrimental to the RL process, and they are present in multiple stages of real-world implementation processes. Three different types of delay are recognized in MDPs [41]:

1. Observation delay, when the state information is not instantly available.

2. Control delay, when the control action is not reflected immediately.
3. Reward delay, when the reward for taking a control action is not immediately obtained.

If a delay is bigger than one time-step, this violates the Markov property, and therefore MDP should be redefined. Delayed environments are specific types of MDP, with an augmented state space and delayed dynamics. The approach to solve delayed MDP has been studied under the assumption of constant delays [93][58], which is unrealistic in real-world scenarios. Although recent work shows a strong performance under constant and random delays [9], it is unclear how delays impact RLED performance or how demonstration trajectories can be used to improve current methods for delayed environments.

4.2.2 Observability

Another common challenge in real-world systems is partial observability, which refers to potentially important information about the state not directly observable. In the same way as a delayed environment, partial observability is formulated as a specific type of MDP. The dimensionality of the unobserved state-space makes it a complex problem. In the current literature, many algorithms have been proposed to handle partial observable MDPs, such as convolutional layers [61] or recurrent layers [31]. Nevertheless, the RLED capacity to handle partial observability has not been fully explored. Another option to explore is how RLED algorithms can be used to design effective exploration strategies in partial observable MDPs.

4.2.3 Detrimental demonstrations

Harmful demonstrations are likely to occur in real-world scenarios, where an RLED agent could be implemented and trained by any user. Users naturally or even maliciously could provide different types of demonstration trajectories presenting, dangerous behaviors to the environment, confusing behaviors showing very different solutions for the same task (probably from different teachers), and useless behaviors containing no relevant data. It is unclear what might cause these types of detrimental demonstrations in the methods discussed and, more importantly, whether detrimental demonstrations can be used to learn a good control policy.

4.2.4 Accelerating RL process with demonstrations

By adding demonstration trajectories to the RL process, one additional hyperparameter could be considered, the amount of relevant data needed to accelerate the learning process and obtain the desired behavior. It is not clear how much relevant data a single demonstration should contain, and given this, how many demonstration trajectories should be provided. It certainly would be beneficial to know how much we can accelerate the RL process with a set of demonstrations containing a certain amount of relevant data.

Approach	Assumptions				Objectives
	R	**EI**	**PK**	**OK**	
RL	✓	✓	-	-	To find a policy that maximizes the reward by interacting with the environment.
RLED with prior knowledge	✓	✓	✓	-	To find a policy that maximizes the reward by interacting with the environment and evaluating prior demonstrations.
Inverse RL	-	✓	✓	-	To find a policy that maximizes an inferred reward from demonstrations by interacting with the environment.
Batch RL	✓	-	✓	-	To find a policy that maximizes the reward by evaluating prior knowledge
RLED with online knowledge	✓	✓	-	✓	To find a policy that maximizes the reward by interacting with the environment and evaluating online knowledge.
Human evaluative feedback RL	-	✓	-	✓	To find a policy that maximally complies with human scalar feedback by interacting with the environment
Human preferences RL	-	✓	-	✓	To find a policy that maximally complies with human preferences by interacting with the environment
Human attention RL	-	✓	-	✓	To find a policy that maximally follows an attention human map by interacting with the environment
Hierarchical imitation RL	-	✓	✓	✓	To find a high-level policy that achieves high-level goals when the high and low policies run together by following demonstrations, querying the human for the correct goals and actions y interacting with the envirionment
Preferences + IL	-	✓	✓	✓	To find a policy that maximally complies with human preferences and demonstrations by interacting with the environment

Table 2 Comparisions of RLED and other RL approaches. **R**: Available environment reward; **EI**: Available online interactions with the environment; **PK**: Available previous knowledge; **OK**: Available online knowledge.

4.3 Other forms of human guidance methods

Human-guided RL is not only limited to approaches like IRL, Batch RL, or RLED. Other approaches can provide information to guide the RL process, as mentioned by Taylor [86]. Other forms of human in the loop can be used as guidance for RL, where the knowledge is available through a human in the online learning process, which can be seen as feedback. This kind of guidance is beneficial when the task is hard to execute by a human. Therefore, demonstra-

tions might not provide a decent knowledge source, yet humans can provide helpful feedback regarding the performance and guide the agent's learning process. Zhang et al. [102] consider other human guidance approaches, different from knowledge acquired by demonstrations. These other human guidance approaches include human evaluative feedback, human preferences, hierarchical imitation, human attention, and state sequences without actions.

Combining multiple sources of knowledge is a promising approach that could help train sufficiently robust RL algorithms. Ibarz et al. [35] propose a method to train a reward model; first, the agent learns from demonstrations using the pre-training phase of DQfD, later, using human preferences [13] and demonstrations. Unlike the common RLED approach, it is assumed that the reward is not available. Instead, it is assumed that a human in the loop is available who has an intention for the agent's task. The human in the loop communicates its intention for the agent's task through demonstrations and preferences that will be used to infer a reward function. This approach guides the RL agent when an explicit reward (sparse or dense) is not available.

5 Conclusions

Reinforcement Learning is an area that is growing at an ever-increasing rate with more innovative ideas and approaches. However, there is still much room for improvement to take artificial intelligence to the next level. This article summarizes the most relevant methods of RLED, which is a promising approach to behavioral learning through demonstrations from an expert teacher. Also, we review the general assumptions that mainly identify this approach and propose a classification for this set of methods.

RLED is an area with many open research opportunities, as real-world challenges remain. Some RLED methods are sufficiently general in the use of demonstration trajectories and have a high potential to be adapted to other approaches and increasingly improved. However, it is not yet clear and has not been fully explored under what circumstances it is possible to adapt different RLED methods to other approaches.

In the future, more advanced RLED methods may emerge that can face more complex and challenging environments (under situations such as delays, noise, or partial observations) with sufficient robustness. New ideas will generate novel and exciting classifications in the RLED approach, thus changing the use of demonstrations and the base methods. An exciting branch to investigate would be how an RL agent can benefit from using demonstrations that could be considered detrimental at first sight. Another possible branch could involve combining knowledge from different sources, not just from demonstrations, to more easily train control policies to describe a sufficiently robust desired behavior.

Acknowledgment

This work was supported in part by CONACYT under Grant CONACyT-A1-S-8216, by CINVESTAV under Grant SEP-CINVESTAV-62 and Grant CNR-CINVESTAV.

References

1. Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., Kumar, V.: Robel: Robotics benchmarks for learning with low-cost robots. In: Conference on Robot Learning, pp. 1300–1313. PMLR (2020)
2. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems **57**(5), 469–483 (2009)
3. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017)
4. Banerjee, B., Vittanala, S., Taylor, M.E.: Team learning from human demonstration with coordination confidence. The Knowledge Engineering Review **34** (2019)
5. Bellman, R.: On the theory of dynamic programming. Proceedings of the National Academy of Sciences of the United States of America **38**(8), 716 (1952)
6. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
7. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1798–1828 (2013)
8. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Handbook of robotics chapter 59: Robot programming by demonstration. Handbook of Robotics. Springer (2008)
9. Bouteiller, Y., Ramstedt, S., Beltrame, G., Pal, C., Binas, J.: Reinforcement learning with random delays. In: International Conference on Learning Representations (2021)
10. Brys, T., Harutyunyan, A., Suay, H.B., Chernova, S., Taylor, M.E., Nowé, A.: Reinforcement learning from demonstration through shaping. In: Twenty-fourth international joint conference on artificial intelligence (2015)
11. Chemali, J., Lazaric, A.: Direct policy iteration with demonstrations. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
12. Chen, S.A., Tangkaratt, V., Lin, H.T., Sugiyama, M.: Active deep q-learning with demonstration. Machine Learning pp. 1–27 (2019)
13. Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. In: Advances in Neural Information Processing Systems, pp. 4299–4307 (2017)
14. Cools, S.B., Gershenson, C., D'Hooghe, B.: Self-organizing traffic lights: A realistic simulation. In: Advances in applied self-organizing systems, pp. 45–55. Springer (2013)
15. Cruz Jr, G.V., Du, Y., Taylor, M.E.: Pre-training neural networks with human demonstrations for deep reinforcement learning. Workshop on Adaptive and Learning Agents (ALA) at the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2018)
16. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: An empirical investigation of the challenges of real-world reinforcement learning. arXiv preprint arXiv:2003.11881 (2020)
17. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. Nature **590**(7847), 580–586 (2021)
18. Fortunato, M., Azar, M.G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., et al.: Noisy networks for exploration. In: International Conference on Learning Representations (2018)
19. Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., Darrell, T.: Reinforcement learning from imperfect demonstrations. arXiv preprint arXiv:1802.05313 (2018)
20. Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1), 1437–1480 (2015)

21. Garmulewicz, M., Michalewski, H., Miłoś, P.: Expert-augmented actor-critic for vizdoom and montezumas revenge. arXiv preprint arXiv:1809.03447 (2018)
22. Gimelfarb, M., Sanner, S., Lee, C.G.: Reinforcement learning with multiple experts: A bayesian model combination approach. In: Advances in Neural Information Processing Systems, pp. 9528–9538 (2018)
23. Goecks, V.G., Gremillion, G.M., Lawhern, V.J., Valasek, J., Waytowich, N.R.: Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 465–473 (2020)
24. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
25. Gordon-Hall, G., Gorinski, P.J., Cohen, S.B.: Learning dialog policies from weak demonstrations. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1394–1405. Association for Computational Linguistics (2020)
26. Gordon-Hall, G., Gorinski, P.J., Lampouras, G., Iacobacci, I.: Show us the way: Learning to manage dialog from demonstrations. In: The Eight Dialog System Technology Challenge (DSTC-8) at AAAI 2020 (2020)
27. Gulcehre, C., Le Paine, T., Shahriari, B., Denil, M., Hoffman, M., Soyer, H., Tanburn, R., Kapturowski, S., Rabinowitz, N., Williams, D., et al.: Making efficient use of demonstrations to solve hard exploration problems. In: International Conference on Learning Representations (2019)
28. Guss, W.H., Codel, C., Hofmann, K., Houghton, B., Kuno, N., Milani, S., Mohanty, S., Liebana, D.P., Salakhutdinov, R., Topin, N., et al.: The minerl competition on sample efficient reinforcement learning using human priors. NeurIPS Competition Track (2019)
29. Guss, W.H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., Salakhutdinov, R.: Minerl: A large-scale dataset of minecraft demonstrations. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (2019)
30. Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policies. In: International conference on machine learning, pp. 1352–1361. PMLR (2017)
31. Hausknecht, M., Stone, P.: Deep recurrent q-learning for partially observable mdps. In: AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15) (2015)
32. Heess, N., Hunt, J.J., Lillicrap, T.P., Silver, D.: Memory-based control with recurrent neural networks. NIPS Workshop on Deep Reinforcement Learning (2015)
33. Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al.: Deep q-learning from demonstrations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
34. Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., Silver, D.: Distributed prioritized experience replay. In: International Conference on Learning Representations (2018)
35. Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., Amodei, D.: Reward learning from human preferences and demonstrations in atari. In: Advances in Neural Information Processing Systems, pp. 8011–8023 (2018)
36. Jing, M., Ma, X., Huang, W., Sun, F., Yang, C., Fang, B., Liu, H.: Reinforcement learning from imperfect demonstrations under soft expert guidance. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5109–5116 (2020)
37. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The malmo platform for artificial intelligence experimentation. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 4246–4247 (2016)
38. Kakade, S.M.: A natural policy gradient. In: Advances in neural information processing systems, pp. 1531–1538 (2002)
39. Kang, B., Jie, Z., Feng, J.: Policy optimization with demonstrations. In: International Conference on Machine Learning, pp. 2469–2478 (2018)
40. Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., Dabney, W.: Recurrent experience replay in distributed reinforcement learning. In: International Conference on Learning Representations (2018)

41. Katsikopoulos, K.V., Engelbrecht, S.E.: Markov decision processes with delays and asynchronous cost collection. IEEE transactions on automatic control **48**(4), 568–574 (2003)
42. Keller, B., Draelos, M., Zhou, K., Qian, R., Kuo, A.N., Konidaris, G., Hauser, K., Izatt, J.A.: Optical coherence tomography-guided robotic ophthalmic microsurgery via reinforcement learning from demonstration. IEEE Transactions on Robotics **36**(4), 1207–1218 (2020)
43. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research **32**(11), 1238–1274 (2013)
44. Kormushev, P., Calinon, S., Caldwell, D.G.: Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. Advanced Robotics **25**(5), 581–603 (2011)
45. Kormushev, P., Calinon, S., Caldwell, D.G.: Reinforcement learning in robotics: Applications and real-world challenges. Robotics **2**(3), 122–148 (2013)
46. Kurin, V., Nowozin, S., Hofmann, K., Beyer, L., Leibe, B.: The atari grand challenge dataset. arXiv preprint arXiv:1705.10998 (2017)
47. Lakshminarayanan, A.S., Ozair, S., Bengio, Y.: Reinforcement learning with few expert demonstrations. In: NIPS Workshop on Deep Learning for Action and Interaction, vol. 2016 (2016)
48. Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: Reinforcement learning, pp. 45–73. Springer (2012)
49. Lazaric, A., Ghavamzadeh, M., Munos, R.: Analysis of classification-based policy iteration algorithms. The Journal of Machine Learning Research **17**(1), 583–612 (2016)
50. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643 (2020)
51. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. IEEE Control Systems Magazine **32**(6), 76–105 (2012)
52. Li, M., Brys, T., Kudenko, D.: Introspective q-learning and learning from demonstration. The Knowledge Engineering Review **34** (2019)
53. Li, Y., Kash, I., Hofmann, K.: Learning good policies from suboptimal demonstrations. In: 14th European Workshop on Reinforcement Learning (EWRL 2018), vol. 2 (2018)
54. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. International Conference on Learning Representations (2016)
55. Liu, J., Zhang, Y., Wang, X., Deng, Y., Wu, X.: Dynamic pricing on e-commerce platform with deep reinforcement learning. arXiv preprint arXiv:1912.02572 (2019)
56. Liu, Y., Liu, Q., Zhao, H., Pan, Z., Liu, C.: Adaptive quantitative trading: an imitative deep reinforcement learning approach. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 2128–2135 (2020)
57. Lozano-Perez, T.: Robot programming. Proceedings of the IEEE **71**(7), 821–841 (1983)
58. Mahmood, A.R., Korenkevych, D., Komer, B.J., Bergstra, J.: Setting up a reinforcement learning task with a real-world robot. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4635–4640. IEEE (2018)
59. Milani, S., Topin, N., Houghton, B., Guss, W.H., Mohanty, S.P., Vinyals, O., Kuno, N.S.: The minerl competition on sample-efficient reinforcement learning using human priors: A retrospective. Journal of Machine Learning Research **1**, 1–10 (2020)
60. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning, pp. 1928–1937 (2016)
61. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
62. Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Overcoming exploration in reinforcement learning with demonstrations. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6292–6299. IEEE (2018)
63. Ng, A.Y., Russell, S.J.: Algorithms for inverse reinforcement learning. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 663–670 (2000)

64. Osband, I., Blundell, C., Pritzel, A., Roy, B.V.: Deep exploration via bootstrapped dqn. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 4033–4041 (2016)
65. Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., Andrychowicz, M.: Parameter space noise for exploration. In: International Conference on Learning Representations (2018)
66. Pohlen, T., Piot, B., Hester, T., Azar, M.G., Horgan, D., Budden, D., Barth-Maron, G., Van Hasselt, H., Quan, J., Večerík, M., et al.: Observe and look further: Achieving consistent performance on atari. arXiv preprint arXiv:1805.11593 (2018)
67. Powell, W.B.: Approximate Dynamic Programming: Solving the curses of dimensionality, vol. 703. John Wiley & Sons (2007)
68. Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., Levine, S.: Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In: Robotics: Science and System XIV (2018)
69. Ravichandar, H., Polydoros, A.S., Chernova, S., Billard, A.: Recent advances in robot learning from demonstration. Annual Review of Control, Robotics, and Autonomous Systems **3** (2020)
70. Resnick, C., Raileanu, R., Kapoor, S., Peysakhovich, A., Cho, K., Bruna, J.: Backplay:" man muss immer umkehren". In: Workshop on Reinforcement Learning in Games at AAAI-19 (2019)
71. Rigter, M., Lacerda, B., Hawes, N.: A framework for learning from demonstration with minimal human effort. IEEE Robotics and Automation Letters **5**(2), 2023–2030 (2020)
72. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems, vol. 37. University of Cambridge, Department of Engineering Cambridge, UK (1994)
73. Salimans, T., Chen, R.: Learning montezuma's revenge from a single demonstration. arXiv preprint arXiv:1812.03381 (2018)
74. Schaal, S.: Learning from demonstration. In: Advances in neural information processing systems, pp. 1040–1046 (1997)
75. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: International Conference on Learning Representations (2016)
76. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning, pp. 1889–1897 (2015)
77. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
78. Segre, A., DeJong, G.: Explanation-based manipulator learning: Acquisition of planning ability through observation. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation, vol. 2, pp. 555–560. IEEE (1985)
79. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
80. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: International conference on machine learning, pp. 387–395. PMLR (2014)
81. Suay, H.B., Brys, T., Taylor, M.E., Chernova, S.: Learning from demonstration for shaping through inverse reinforcement learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp. 429–437 (2016)
82. Subramanian, K., Isbell Jr, C.L., Thomaz, A.L.: Exploration from demonstration for interactive reinforcement learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp. 447–456 (2016)
83. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
84. Sutton, R.S., Barto, A.G., Williams, R.J.: Reinforcement learning is direct adaptive optimal control. IEEE Control Systems Magazine **12**(2), 19–22 (1992)
85. Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning, vol. 135. MIT press Cambridge (1998)
86. Taylor, M.E.: Improving reinforcement learning with human input. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 5724–5728 (2018)
87. Taylor, M.E., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th international conference on Machine learning, pp. 879–886 (2007)

88. Taylor, M.E., Suay, H.B., Chernova, S.: Integrating reinforcement learning with human demonstrations of varying ability. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 617–624. International Foundation for Autonomous Agents and Multiagent Systems (2011)
89. Torrey, L.: Reinforcement learning via reasoning from demonstration. Workshop on Adaptive and Learning Agents (ALA) at the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2020)
90. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
91. Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., Riedmiller, M.: Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:1707.08817 (2017)
92. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)
93. Walsh, T.J., Nouri, A., Li, L., Littman, M.L.: Learning and planning in environments with delayed feedback. Autonomous Agents and Multi-Agent Systems **18**(1), 83–105 (2009)
94. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International conference on machine learning, pp. 1995–2003. PMLR (2016)
95. Wang, Z., Taylor, M.E.: Improving reinforcement learning with confidence-based demonstrations. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3027–3033 (2017)
96. Wang, Z., Taylor, M.E.: Interactive reinforcement learning with dynamic reuse of prior knowledge from human and agent demonstrations. In: Twenty-Eighth International Joint Conference on Artificial Intelligence (2019)
97. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)
98. Wu, X., Zhang, D., Qin, F., Xu, D.: Deep reinforcement learning of robotic precision insertion skill accelerated by demonstrations. In: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pp. 1651–1656. IEEE (2019)
99. Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In: Advances in neural information processing systems, pp. 5279–5288 (2017)
100. Yeo, S., Oh, S., Lee, M.: Accelerating deep reinforcement learning using human demonstration data based on dual replay buffer management and online frame skipping. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 1–8. IEEE (2019)
101. Yeo, S., Oh, S., Lee, M.: Accelerated deep reinforcement learning with efficient demonstration utilization techniques. World Wide Web pp. 1–23 (2020)
102. Zhang, R., Torabi, F., Guan, L., Ballard, D.H., Stone, P.: Leveraging human guidance for deep reinforcement learning tasks. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (2019)
103. Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., Kumar, V.: Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 3651–3657. IEEE (2019)
104. Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., Levine, S.: The ingredients of real-world robotic reinforcement learning. In: International Conference on Learning Representations (2020)

https://dspace.org/

Model-free reinforcement learning from expert demonstrations: a survey

Ramírez, Jorge; Yu, Wen; Perrusquía, Adolfo

2021-10-18
Springer
https://doi.org/10.1007/s10462-021-10085-1
Ramirez J, Yu W, Perrusquia A. (2022) Model-free reinforcement learning from expert demonstrations: a survey. Artificial Intelligence Review, Volume 55, Issue 4, April 2022, pp. 3213-3241

Downloaded from a DSpace repository. https://dspace.org/