Autonomous Agents and Multi-Agent Systems (2021) 35:44
https://doi.org/10.1007/s10458-021-09527-5

Goal-driven active learning

Nicolas Bougie[1,2] · **Ryutaro Ichise**[1,2]

Accepted: 28 July 2021 / Published online: 16 August 2021

Abstract
Deep reinforcement learning methods have achieved significant successes in complex decision-making problems. In fact, they traditionally rely on well-designed extrinsic rewards, which limits their applicability to many real-world tasks where rewards are naturally sparse. While cloning behaviors provided by an expert is a promising approach to the exploration problem, learning from a fixed set of demonstrations may be impracticable due to lack of state coverage or distribution mismatch—when the learner's goal deviates from the demonstrated behaviors. Besides, we are interested in learning how to reach a wide range of goals from the same set of demonstrations. In this work we propose a novel goal-conditioned method that leverages very small sets of goal-driven demonstrations to massively accelerate the learning process. Crucially, we introduce the concept of active goal-driven demonstrations to query the demonstrator only in hard-to-learn and uncertain regions of the state space. We further present a strategy for prioritizing sampling of goals where the disagreement between the expert and the policy is maximized. We evaluate our method on a variety of benchmark environments from the Mujoco domain. Experimental results show that our method outperforms prior imitation learning approaches in most of the tasks in terms of exploration efficiency and average scores.

Keywords Deep reinforcement learning · Imitation learning · Goal-conditioned learning · Active learning

1 Introduction

Recent successes in deep reinforcement learning (DRL) have been achieved in domains with a well-specified reward function such as in game-playing [53] or robot control [49]. Unfortunately, many real-world tasks involve rewards that are poorly-defined, sparse, or delayed. Moreover, these algorithms typically require a large number of interactions to

✉ Nicolas Bougie
nicolas-bougie@nii.ac.jp

Ryutaro Ichise
ichise@nii.ac.jp

[1] The Graduate University for Advanced Studies (Sokendai), Tokyo, Japan

[2] National Institute of Informatics, Tokyo, Japan

reach decent performance, which can be intractable in real-world settings. Overcoming these pitfalls could help to expand the possible applications of DRL.

A line of work for overcoming the above-mentioned issues is goal-conditioned learning, a form of self-supervision that constructs a goal-conditioned policy to learn how to reach multiple goals [44, 68]. This idea was extended in Hindsight Experience Replay (HER) [4] to artificially generate new transitions by relabeling goals seen along the state trajectory. Nevertheless, it may still require a large amount of data to capture complex policies. Since it is often unrealistic to expect an end-to-end reinforcement learning system to rapidly succeed with no prior assumptions about the domain (i.e. learning a task from scratch), several methods have attempted to introduce external supervision into reinforcement learning systems. For instance, an approach [39] leverages human preferences as feedback signal. Nonetheless, it was shown that preferences are an inefficient way of soliciting information from humans [39]. In the context of reinforcement learning, the most common form of external supervision is imitation learning. Imitation learning seeks to learn tasks from demonstrated state-action trajectories [1, 67]. For instance, Deep Q-learning from Demonstrations (DQfD) [34] improves initial performance by pre-training the policy with demonstrations. However, learning from human demonstrations suffers from three problems: (1) it is hard to obtain a broad state coverage of task-relevant regions from trajectories demonstrated without a specific goal, (2) it usually has an abundance of irrelevant or redundant information, (3) it assumes that the learner's goal matches the teacher's demonstrated behaviors. Additionally, most imitation learning algorithms learn policies that achieve a single task.

In this work, we contribute an active goal-conditioned approach that drastically reduces expert workload by incrementally requesting partial demonstrations towards specific goals, *goal-driven demonstrations*. Contrary to pure demonstrations, goal-driven demonstrations do not aim to demonstrate the overall task or all possible situations. Instead, goal-driven demonstrations fulfill particular goals that are actively selected based on the agent's knowledge about its environment. Especially, the proposed framework allows an agent to jointly identify states where feedback is most needed and communicate for specific domain knowledge throughout the training process. Our method relies on an *imitator* network trained to clone a novel form of human feedback: *goal-driven demonstrations*. Given its prediction, we augment the policy loss with a simple auxiliary objective. Rather than using a fixed set of demonstrations, goal-driven demonstrations are actively queried based on the imitator's confidence and the ability of the agent to reach the goal being pursued. We build and compare two techniques to estimate the agent's confidence: (1) Bayesian-confidence, (2) quantile-confidence; and study a relabeling strategy that extracts additional information from the demonstrated trajectories. We found goal-driven demonstrations to be easier to demonstrate for a human than full demonstrations, while significantly increasing the value information of the queries by matching the agent's needs. We further propose a method for prioritizing the sampling of important goals—in places where the disagreement between the expert and the policy is large.

We evaluate our approach on several tasks from the Mujoco benchmark suite [61, 77] including Fetch and ShadowHand. Experimental results show that GoAL outperforms previous approaches in most of the tasks with a significantly lower number of demonstrations. We also show that our method can generalize to unseen states while being robust to incomplete or noisy demonstrations. Remarkably, GoAL produced agents that exceeded the expert performance in multiple tasks.

The main contributions of this paper are summarized as follows:

- We propose a new framework, Goal-driven Active Learning (GoAL), which is the first work to use active goal-driven demonstrations to the best of our knowledge.
- We contribute a method to query the demonstrator only in states where the agent struggles and is not confident, maximizing the expected value of information of the queries and drastically reducing human effort.
- We propose two novel confidence-based query strategies to evaluate the confidence along a state-action trajectory.
- We contribute a goal-sampling technique that maximizes the agent's learning progress.
- We provide a comprehensive comparison between the proposed methodology and a number of baselines, evaluated on complex robotic tasks.

The remainder of the paper is organized as follows. Section 2 reviews related literature. Section 3 provides the necessary background to the research topics presented in the paper. Section 4 details the description of the proposed algorithm, and Sect. 5 reviews the experiments to verify the algorithm. Finally, Sect. 6 provides a summary and suggests future research, and Sect. 7 concludes this work.

2 Related work

Learning when rewards are sparse is a notoriously challenging problem in the field of reinforcement learning. One solution to tackle sparse rewards is to introduce an intrinsic incentive into reinforcement learning, curiosity. On the other hand, methods for providing external supervision largely divide into two categories: imitation learning and learning from interactive human feedback. Our work is built upon goal-conditioned learning. We briefly introduce these techniques in this section.

2.1 Curiosity-driven exploration

Inspired by curious behavior in animals, the use of intrinsic motivation has been developed to encourage agents to learn about their environments even when extrinsic feedback is rarely provided. Some techniques [59, 71] rely on predicting environment dynamics using an inverse or forward dynamic model. Another class of approaches uses prediction errors in the feature space as measure of the importance of states [47]. For example, RND [11] predicts the output of a randomly initialized neural network on the current state, and encourages revisits of states with large prediction errors. Episodic curiosity through reachability [66] addresses the "noisy TV" issue of prior work by considering the distance between two states as curiosity measure. Exploration bonus can also be based on maximizing information gain about the agent's knowledge of the environment [36]. In GoCu [9] curiosity is formulated as the capability of the agent to learn a set of skills. Another line of work is to keep visit counts for states to favor exploration of rarely visited states [7, 50, 72]. In order to enable count-based exploration in continuous state spaces, a solution [57] is to train an observation density model to supply counts. Another strategy [75] is to map states to hash codes and count state visitations with a hash table. In this setting, the counts are used as exploration bonus to guide exploration. A prior work [51] introduces a count-based optimistic algorithm by estimating the uncertainty associated with each state. In a slightly different spirit, DIAYN [21] proposes to learn useful skills without a reward function—they learn skills by optimizing an information theoretic objective using a maximum entropy

policy. While curiosity was shown to be useful when rewards are sparse, training an end-to-end reinforcement learning system with no prior assumptions about the domain often requires millions or billions of interactions to reach reasonable performance, which can be impractical in real-world settings. On the other hand, our work leverages small amounts of human feedback to massively accelerate the learning process. Besides, we are often interested in learning to reach a wide range of goals without re-training the agent or designing a different reward every time. This capability is essential in many real-world domains such as robot control.

2.2 Imitation learning (IL) and RL

One of the earliest attempts at end-to-end behavioral cloning was ALVINN [62], for lateral motion control of an autonomous vehicle. In recent years, multiple work have attempted to combine deep reinforcement learning with human demonstrations. For instance, DQfD [34] pre-trains a Q-learning agent on the expert demonstration data. This idea was extended to handle continuous action spaces such as in robotic tasks [78], as well as to actor-critic architectures [84]. POfD [45] proposes to follow demonstrations in early learning stages for exploration and let the agent explore new states on its own. In contrast, we are interested in actively sharing insights between the teacher and the agent. Therefore, we propose a novel imitation loss function that leverages goal-driven demonstrations and a goal-conditioned framework to actively request feedback to the teacher when the agent struggles, reducing both the training time and the number of demonstrations [18]. A recent follow-up [54] introduces an expert loss in DDPG [49] and proposes to filter suboptimal demonstrations based on the Q-values (Q-filter). It is assumed that there is a fixed set of demonstration data. In this work, we use the Q-filter method [54] in a goal-conditioned setting, and we further adapt it to filter transitions where the agent action is significantly better than the demonstrator action. Another solution is to represent a policy as a set of Gaussian mixture models [16]. However, they consider a fixed target goal setting, and the method is not directly applicable to continuous action spaces. In a different spirit, AlphaGo [70] trains a policy network to classify positions according to expert moves. A way of dealing with sparse rewards consists in introducing a curious replay mechanism and demonstrations [86]. DAGGER [48] requests supervision at each step and takes an action sampled from a mixture distribution of the demonstrator and the agent. The idea was extended in Deeply AggreVaTeD [73] to work in environments with continuous action spaces. Another method [19] constructs a goal-conditioned policy to visit similar states as the expert. That is, they employ the idea of discriminability as a central theme in building agents that can leverage demonstrations. Rather than solely using goals to condition the policy, we use goals to enable active cooperation between the teacher and the agent. Namely, we propose a novel form of human guidance, *goal-driven demonstrations*. Goal-driven demonstrations do not intend to cover all possible scenarios or demonstrate the overall task, but guide the agent to fulfill particular goals when the agent struggles, being more intuitive for the demonstrator than pure demonstrations. In addition, the imitation loss is used in a different way in our method; and we develop a different strategy for relabeling goal-driven demonstrations, which ensures that only optimal transitions are recorded. Another form of imitation learning is inverse reinforcement learning (IRL) [56] where a reward function is inferred from the demonstrated trajectories. IRL has been applied to several domains including navigation [5], and autonomous flight [1]. In recent years, an emerging strategy at the intersection of imitation learning and IRL has combined generative adversarial networks and

reinforcement learning (GAIL) [35]. However, IRL algorithms assume that the observed behavior is optimal, and most agents focus on learning a single task from a set of demonstrations. Another issue concerns IL and IRL approaches that leverage expert rewards (e.g. demonstrations). In many cases, it is impractical to generate large amounts of high-quality demonstrations, especially for long-term tasks. In order to practically train RL systems with human feedback, we need to decrease the amount of human effort. Many work in the imitation learning literature assume that an expert cannot be queried because it is impractical and costly. On the other hand, we argue that querying an expert can be a strength since it allows us to reduce the number of necessary demonstrations by adapting demonstration data to match the agent's needs, ultimately reducing human effort. A constant supervision is very impractical, so our method lets the agent identify querying opportunities so that the expert is not required to constantly monitor the agent. In order to identify the need for specific domain knowledge throughout the agent's training, we contribute a framework to identify areas where feedback is most needed based on the agent's confidence and its ability to reach the goal pursued. On the other hand, in the absence of active cooperation, it is often challenging for an expert to know in advance what will be the agent's needs.

2.3 Learning from interactive human feedback

Most methods that focus on learning from interactive human feedback [17, 39, 52, 80, 81] query the human to drive learning [18]. For example, TAMER [79] trains the policy from feedback in high-dimensional state space. The learner may receive feedback in the form of sequences of actions planned by a teacher [10]. Uncertainty-based query was used in [14] but is limited to DQN [53], limiting the possible applications of this method. In contrast, our method can be combined with most of off-policy RL algorithms; and introduces the idea of goal-driven demonstrations. Some authors [69] consider multiple demonstrators performing different tasks and the agent must actively select which one to request for advice. Another solution [65] is to block unsafe actions by training a module from expert feedback. However, it requires the expert to identify all unsafe situations by watching an agent play. To deal with the problem of query selection, it is possible to select sufficiently different unqueried data [37]. In a similar spirit, algorithms in the field of action advising aim to transfer action advice under a budget from a teacher to the agent [22]. For instance, in L2T [23] a teacher model leverages the feedback from the student model in order to optimize its own teaching strategies, achieving teacher-student co-evolution. Despite L2T performance on image classification and sentiment analysis, it remains unclear how to apply this approach to more complex environments. Another algorithm [85] consists in letting the student announce his recommended action and the teacher can decide whether to provide some advice. In the same spirit, a work has considered multiple teaching algorithms [76] and applied them to game-playing. While these methods effectively accelerate agent training, they assume that the teacher constantly monitors the agent. This assumption may not hold with human teachers, as humans have temporally limited attention, and the cost of monitoring may be prohibitive. To overcome these pitfalls, it is possible to identify advising opportunities so that the teacher is not required to constantly monitor the student [3]. These approaches assume that a piece of advice consists in suggesting the action that the student should do. Our work differs by leveraging goal-driven demonstrations, allowing us to transfer more complex domain knowledge and removing the need for constant monitoring. In addition, we propose a strategy to only request feedback to the supervisor in states where the agent is unsure and struggles.

2.4 Curriculum learning

The idea of *active queries* to an expert is also closely related to the field of curriculum learning [8]. Unlike machine learning, human learning is often accompanied by a curriculum. That is, the order of presented examples is rarely random when a human teacher teaches another human. One popular approach is to decide which task to solve next based on the agent's learning progress [58]. This strategy can be extended to consider learning progress in terms of rate of increase in prediction accuracy and rate of increase in network complexity [33]. A recent follow-up [6] proposed a novel curriculum generation method using different progression functions, including a function based on the performance of the agent. The authors also use the progression function to determine how long the agent should train on each intermediate task. In order to prevent forgetting of earlier tasks, a probability of returning to earlier tasks can be defined [83]. Another approach relies on providing to the agent increasingly difficult goals [26]. A few studies have considered setter-solver paradigm; e.g. [64] considered goal feasibility and goal coverage to construct curricula. In the proposed method, we assign high sampling priority to goals where the expert and the agent strongly disagree. Another form of curriculum learning is PLO that constructs auxiliary policies that learn from shaped reward functions, allowing the main policy to gradually get more independent and execute more actions sampled from its own policy [38]. In order to learn from sparse rewards, one solution is to use curriculum learning that breaks a complex task into sub-tasks of gradually increasing complexity and learning them concurrently [2]. The presented work forms an implicit curriculum by gradually querying more complex goal-driven demonstrations as the agent's knowledge about the environment increases. It can be combined with an arbitrary off-policy RL algorithm and may be seen as a form of implicit curriculum. Our method is based on hindsight experience replay [4] that may also be seen as a form of implicit curriculum. The central idea is to replay each episode by replacing the desired goals of training trajectories with the achieved goals of the failed experience.

2.5 Few-shot imitation learning

Few-shot imitation learning was proposed as a way to leverage a few demonstrations of a certain task, and have these demonstrations instantly generalize to new situations of the same task. For instance, a work aims to maximize the expected performance of the learned policy when facing a new task, without receiving additional demonstrations [20]. In order to reduce the number of demonstrations needed for an individual task, a strategy is to share data across tasks and learn a parameterized policy that can be adapted to different tasks through gradient updates [25]. Another related work is MAML [24], where the agent learns a set of weights that can be quickly adapted to new tasks from one visual demonstration. Another approach employs ideas from metric learning in order to learn a task embedding that can be used to learn new tasks from a few demonstrations [40]. In this work, we focus on settings where the agent learns a single task with multiple goals. Moreover, our approach is centered around the idea that learning efficiency can be greatly improved by allowing active cooperation between the agent and a teacher.

2.6 Goal-conditioned RL

Goal-conditioned reinforcement learning [44] constructs a goal-conditioned policy to push the agent to acquire new skills and explore novel states. Universal value function approximators [68] sample a fixed goal at the beginning of each episode and reward the agent when the current goal can be achieved. Nonetheless, selecting relevant goals remains an open problem. A solution [26] and its recent follow-up [55], proposed to generate increasingly difficult goals to drive the agent towards the final goal. The method [60] learns an embedding for the goal space using unsupervised learning and then choose the goals from that space. The recent work [63], Skew-fit, proposes an exploration objective that maximizes state diversity. The key idea is to learn a maximum-entropy goal distribution to match the weighted empirical distribution, where the rare states receive larger weights. Another line of work [28] focuses on goals that provide maximal learning progress. However, defining *when*, *to whom*, and *how* to ask instructions to the demonstrator remains an open problem [18]. Our method, which builds on top of HER, provides an order of magnitude of speedup by taking advantage of very few goal-driven demonstrations. We further introduce a novel goal sampling strategy based on the disagreement with the demonstrator.

3 Background

In this section, we briefly review the reinforcement learning techniques that our method is built on, including goal-conditioned learning and hindsight experience replay.

We consider a finite-horizon Markov decision process (MDP) as a tuple (S, A, P, r, γ), where S is a set of states, A is a set of possible actions, $P : S \times A \times S \to \mathbb{R}$ is a transition function, $r : S \times A \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. We aim to find a policy $\pi : S \to A$ that maximizes the expected discounted reward, $R_t = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, s_{t+1})]$.

In this work, we use a goal-conditioned formulation where the reward function and the policy are additionally conditioned on a goal, $g \in G$. At every timestep the agent gets as input not only the current state but also the current goal, thus the policy selects the next action given a state and a goal $\pi : S \times G \to A$. The reward function becomes $r_t = r_g(s_t, a_t)$ where r_g is often a binary function which represents whether the agent could reach the goal (i.e. $\mathbb{1}[s_{t+1} == g]$. HER [4] showed that gathered trajectories can be artificially relabeled with new goals. The central idea is to replay each episode with additional goals than the one the agent was trying to reach. Namely, they replay each transition with the original goal pursued in the episode as well as randomly selected goals along the trajectory. Since the transition probability is not affected by the goal being pursued g, the tuple can be relabeled in hindsight. Thus, a transition $(s_t, a_t, s_{t+1}, g, r = 0)$ can be treated as $(s_t, a_t, s_{t+1}, g' = s_{t+1}, r = 1)$. By doing so, it drives the agent to learn how to achieve multiple goals without simulating interactions—generating and recomputing rewards of a single transition can be converted into many valid training examples.

4 Method

Algorithm 1 Goal-driven Active Learning (GoAL)

Given:
- an off-policy RL algorithm π_θ ▷ e.g. DQN, DDPG, NAF
- a replay buffer R and an expert trajectory buffer Ω
- an imitator policy f_ϑ and a query threshold t_{qry}
- a confidence function c
- a reward function $r_g : S \times A \times G \to \mathbb{R}$ ▷ e.g. $r_g(s, a, g) = ||s - g||_2^2$

Initialize π_θ and f_ϑ ▷ initialize neural networks
Initialize $R = \{\}$ and $\Omega = \{\}$
for m=0,...,M episodes **do**
  Receive a goal g and initial state s_0 from the environment
  for t=0,...,H-1 steps **do**
    Get action $a_t \sim \pi(a_t|s_t, g; \theta)$
    Execute a_t and observe next state s_{t+1}
    Store transition $R = R \cup (s_t, a_t, s_{t+1}, g, r)$
  Save the current trajectory $\varphi = \{(s_0, a_0, g), ...\}$ and the final state $s_k = s_{H-1}$
  $C(\varphi) = \mathbb{1}[s_k \neq g]\mathbb{E}_{(s_t,g)\sim\varphi} c(s_t, g)$ ▷ Query Selection
  if $C(\varphi) > t_{qry}$ **then**
    Query the expert with g as the target goal and receive $\tau = \{(s_0, a_0), (s_1, a_1), ..., (s_k, g)\}$ ▷ Query the demonstrator
    Relabel τ and store tuples in Ω ▷ Expert Relabeling
    Fine-tune f_ϑ on Ω
  for t=0,...,H-1 steps **do**
    Sample a set of additional goals $G \in \{s_{t+1}, .., s_{H-1}\}$ with probability $p^g(s)$ ▷ Prioritized Goal Sampling
    for $g' \in G$ **do**
      Store transition $R = R \cup (s_t, a_t, s_{t+1}, g', r_g(s_t, a_t, g'))$
  for j=0,...,N_{opt}-1 **do**
    Sample a minibatch B from R
    Calculate L_C on B with Q-filter ▷ Goal-Driven Imitation Loss
    Update policy loss $L^D = \lambda_1 L + \lambda_2 L_e$ and optimize θ

The challenges of injecting expert feedback into DRL are twofold. First, expert demonstrations are limited, which entails that the agent needs to efficiently leverage a small amount of demonstration data. Although a number of algorithms could in principle be used to learn from demonstrations, standard methods can suffer from poor performance. This can happen when the state coverage of the expert trajectories is too narrow, or due to a discrepancy between the agent's goal and the demonstrated data. Second, demonstrating the entire task trajectory multiple times is an inefficient way of soliciting information from humans, lacking of generalization capability to new target goals.

The framework of *Goal-driven Active Learning* (GoAL) provides us a mechanism to mitigate these problems by incrementally querying *goal-driven* demonstrations (Fig. 1). Our approach (Algorithm 1) introduces human feedback into goal-conditioned learning via Hindsight Experience Replay. Specifically, the agent receives feedback in the form of short goal-driven demonstrations—the tutor is requested to reach a specific goal. We decide how to query goal-driven demonstrations based on the agent's needs and the

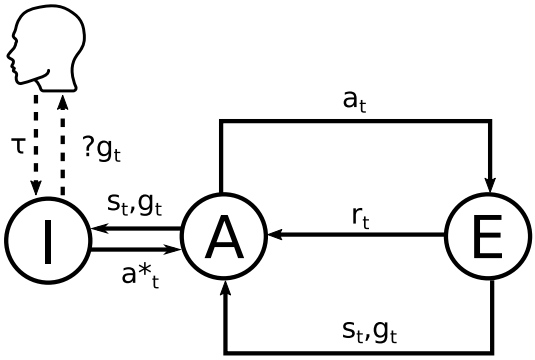


Fig. 1 Goal-driven Active Learning (GoAL). The imitator I predicts the action the demonstrator would have taken given a pair of state and goal (s_t, g_t) provided by the environment E. When the agent A fails to reach the goal being pursued g_t, a new demonstration τ with g_t as the target goal may be queried

expected value of information of the query, drastically reducing the number of required demonstrations.

Our method works as follows (see Algorithm 1). We first collect a trajectory based on the goal being pursued (lines 4–10). Then, the agent decides whether it should query a goal-driven demonstration to the demonstrator (line 11–12). After each query (line 13), we perform *expert relabeling* to artificially generate more expert data (line 14). Expert relabeling is a type of data augmentation on the provided goal-driven demonstrations. As an intuition, if the agent receives a demonstrated trajectory $(s_0, a_0, g), (s_1, a_1, g), \ldots, (s_k, s_k, g)$ with g the goal being pursued, we can relabel transitions with additional goals seen along the state trajectory. For instance, we can add new transitions $(s_0, a_0, s_1), (s_0, a_0, s_2)$ to the expert trajectory buffer. The imitator policy is then trained to imitate the demonstration data (line 15). Then, we augment the policy loss with an extra objective that aims to mimic the demonstrated behaviors (line 20–23). The transitions used to train the policy are generated following a similar strategy as in HER (line 16–19), except that we modified the goal sampling to take advantage of the demonstrations (line 17). This process continues until the task is mastered. In the following section we describe the key components of our method.

4.1 Goal-driven imitation

We assume a small dataset of tuples (s_i, a_i, g_i) extracted from expert trajectories, Ω. A trajectory segment (also called goal-driven demonstration) is a sequence of observations and actions, $\tau = \{(s_0, a_0, g), (s_1, a_1, g), \ldots, (s_k, a_k, g)\}$, where g indicates the goal pursued by the demonstrator. Our method involves an imitator policy $f : S \times G \rightarrow A$ that mimics expert behaviors, parameterized by a set of trainable parameters ϑ. The imitator policy is trained with a regression loss $\bar{L}$: it predicts the action the demonstrator would have taken given a pair of state and goal (s_i, g_i), $a_i^* \sim f(a_i^* | s_i, g_i; \vartheta)$. In the absence of domain knowledge, a general-purpose choice is to train f_ϑ with a regression loss, the mean-squared-error, $\bar{L} = \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} ||a_i^* - a_i||_2^2$.

A contribution of our paper consists in augmenting the policy loss with an extra term to accommodate the goal-driven expert data. Given a minibatch of T transitions, the imitation loss is given by:

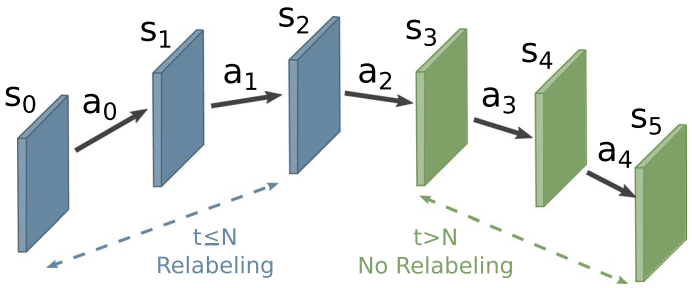


Fig. 2 Relabeling strategy. Given a human trajectory, we artificially generate more expert data by using as the active goal the states within a reachability threshold N. The threshold N is necessary to discard potentially sub-optimal examples (temporally far states)

$$L_e = \frac{1}{T}\sum_{t=1}^{T} ||\pi(a_t|s_t, g_t;\theta) - f(a_t|s_t, g_t;\vartheta)||_2^2 \tag{1}$$

where π is the current policy parameterized by θ. In order to allow the agent to significantly outperform the demonstrator—deviate significantly from the expert demonstrations, we use a Q-filter function [54] in a goal-conditioned setting, which we extend to increase the gap between "optimal" and "sub-optimal" transitions. In order to ensure that a transition provided by the demonstrator is significantly better than the agent's policy, we propose to filter irrelevant transitions via: $\mathbb{1}_{Q(s_t,f(a_t|s_t,g_t),g_t)-(Q(s_t,\pi(a_t|s_t,g_t),g_t)-\eta|Q(s_t,\pi(a_t|s_t,g_t),g_t)|)>0}$, where η is a positive constant. This filtering enables our agent to improve significantly beyond the expert demonstrations (see Sect. 5.4.2), which is especially relevant in the situation where the demonstrators are non-experts or themselves learning the task [18].

The overall loss used to update the policy network is a combination of two losses:

$$L^D = \lambda_1 L + \lambda_2 L_e \tag{2}$$

where L indicates the loss function of any arbitrary DRL algorithms, and λ_1 and λ_2 are hyperparameters to weight the importance of both loss components. Adding this auxiliary objective provides the agent both the intention of the demonstrator and the ability to discover alternative strategies. Next, we show how to artificially increase the amount of demonstrations by relabeling expert data.

4.1.1 Expert relabeling

To further enable sample-efficient learning in the real world, we present a relabeling strategy to artificially generate more expert data. In other words, expert relabeling is a type of data augmentation on the provided goal-driven demonstrations. As mentioned earlier, we collect expert demonstrations in the form of trajectories, $\tau = \{(s_0, a_0, g), (s_1, a_1, g), \ldots, (s_k, s_k, g)\}$, where g is the goal being pursued. The idea behind this method is that in a state s_i, the associated action a_i can be used to reach g, as well as new goals $\{s_{i+1}, \ldots, s_k\}$—the transition probability is not affected by the goal being pursued g. Therefore, we have the freedom to artificially generate more expert data without additional queries, which are referred to as *imaginary samples* since they are imagined by the agent.

In practice, we found that selecting all the future states like done in goalGail [19] not an ideal solution, since distant goals can be reached using different actions. Besides, in the context of active learning, adding imaginary sub-optimal samples may create conflicts when the agent later on receives new feedback from the expert. Instead, we restrict the creation of new imaginary samples to only short slices of the original trajectory. To do so, we propose to use the number of times-steps to approximate the distance between two states. The intuition behind is that temporally far states are more likely to be reached via a different sequence of state-actions than close ones. Thus, the number of times-steps provides a simple strategy to discard sub-optimal samples. We relabel future states when this distance is lower than a threshold N (Fig. 2): $\{s_{i+1}, \ldots, s_{min(i+N,k)}\}$. In our experiments, we found that the GoAL performance is reasonably robust to the choice of this threshold and that N can be simply selected based on the maximum number of time-steps per episode (see Sect. 5.4.1 for more details). By artificially generating new demonstrations, we can convert a single transition (s, g, a) into potentially many valid training examples, which is particularly useful to decrease the number of queries to the demonstrator.

4.2 Query selection

An important component in this method is query selection, in which the agent needs to decide which goals to query for demonstration. Our approach to decide when to query is based on (1) the ability of the agent to reach the goal pursued in the episode, (2) the confidence in the action prediction of the imitator policy. By requesting demonstrations in hard-to-learn and low confidence situations, the depicted algorithm eliminates repetitive demonstrations of already learned goals of the task and provides human feedback to the agent when it struggles.

After experiencing each episode φ, we evaluate the confidence of the imitator along the state-action trajectory, if the goal was failed. For simplicity, a slight abuse of notation is made by using C to denote the query score. A score above a threshold t_{qry} results in a goal-driven query—the demonstrator is requested to demonstrate how to achieve the failed goal. We formally define the overall function to estimate C as:

$$C(\varphi) = \mathbb{1}[s_k \neq g]\mathbb{E}_{(s_t,g)\sim\varphi} c(s_t, g) \tag{3}$$

where s_k is the final state of the trajectory, g is the goal being pursued, and c is an estimation function of the confidence for the pair (s_t, g). Every time a new demonstration is collected, the training transitions are recorded in Ω and we make 50 epochs of training. Rather than using an ensemble-based uncertainty estimate as in prior work—bootstrap samples evaluated by multiple models are used to estimate variance, we propose two novel methods (*quantile-confidence* and *bayesian-confidence*) to estimate prediction confidence, $c(s_t, g)$. In contrast with ensemble-based methods that add a significant overload, the proposed strategies drastically reduce the computational cost. The parametrization is discussed further in the next section.

4.2.1 Quantile-confidence

One common solution for estimating confidence in the prediction relies on ensemble-based uncertainty estimates, as done by Christiano et al. [17]. However, such an approach tends to be computationally expensive [39] and inaccurate when operating in the low data regime (with very few data). Instead, we develop a simple architecture for estimating confidence in

the prediction, which has little/no computational cost, works with most existing imitation-based models, and is more robust against outliers [41, 74]. We propose to embrace deep quantile regression to estimate model confidence. Rather than only predicting the mean, the last layer of f_ϑ is used to predict each quantile separately. Assuming a set of goal-driven demonstration data Ω, we run a regression algorithm to train f_ϑ with the following loss:

$$\bar{L}(q) = \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} \rho(f(a_i|s_i, g_i;\vartheta) - a_i, q) \tag{4}$$

where

$$\rho(\varepsilon, q) = \begin{cases} q\varepsilon, & \text{if } \varepsilon \geq 0 \\ (q-1)\varepsilon, & \text{if } \varepsilon < 0 \end{cases} \tag{5}$$

where q is the required quantile ($0 < q < 1$), and a_i is the action the demonstrator took. We typically use (0.3,0.5,0.8) as quantiles. We can express *quantile-confidence*, $c(s_t, g)$, by measuring the prediction interval between the largest q'' and smallest q' quantile, $c(s_t, g) = \left| f(a_t|s_t, g;\vartheta)_{q''} - f(a_t|s_t, g;\vartheta)_{q'} \right|$. Please note that we use the median quantile (q = 0.5) in Eq. 1.

4.2.2 Bayesian-confidence

Imitator policy confidence can also be modeled using bayesian models. However, in the context of RL, their computational cost can be prohibitive. This problem can be mitigated by using an estimation of Bayesian inference. It was shown that the use of dropout can be interpreted as a Bayesian approximation of Gaussian process [31]. Therefore, we introduce a dropout layer before every weight layer of our imitator policy network. To estimate predictive confidence, we collect the results of stochastic forward passes through the imitator policy network:

$$c(s_t, g) = \mathbb{E}_{d_j \sim D}[f^{d_j}(a_t|s_t, g;\vartheta) - p]^2 \tag{6}$$

where $f^{d_j}(a_t|s_t, g;\vartheta)$ represents the model with dropout mask d_j, D is a set of dropout masks, and p is the predictive posterior mean, $p = \mathbb{E}_{d_j \sim D} f^{d_j}(a_t|s_t, g;\vartheta)$. Since the forward passes can be done concurrently, the method results in a running time identical to that of standard dropout. We can expect the variance of unknown and far-away tuples to be larger than known tuples. Please note that one advantage of using dropout is that it allows the imitator policy to "smooth out" much of the noise in the data, making the imitator policy more robust to noise in the demonstration data. Besides, this technique is particularly effective in the low-data regime to improve generalization of demonstrated behaviors. We compare in Sect. 5.2 the impact of each strategy on our method.

4.3 Prioritized goal sampling

In the future HER sampling strategy, the new goals are randomly selected along the future state-action trajectory. However, an RL agent can learn more effectively from some goals than from others. Typically, some goals may be useful to the agent, but might become less when the agent competence increases. Prioritized goal sampling (PGS) assigns high sampling priority to key goals—in places where the expert and the agent strongly disagree. As

a criterion to quantify this disagreement, we measure the divergence in the action recommendation between the imitator-network and the policy, which indicates how "surprising" or "hard-to-learn" the goal is. Given a state s_t seen along an episode of T states and g the current goal, we define the probability of sampling s_t as a new goal:

$$p^g(s_t) = \frac{||\pi(a_t|s_t, g;\theta) - f(a_t|s_t, g;\vartheta)||_2^2}{\sum_{j=t}^{T} ||\pi(a_j|s_j, g;\theta) - f(a_j|s_j, g;\vartheta)||_2^2} \tag{7}$$

where $||\pi(a_t|s_t, g;\theta) - f(a_t|s_t, g;\vartheta)||_2^2$ is the deviation between the policy and expert. As a result, PGS capitalizes on large disagreement to encourage sampling of goals that potentially lead to large learning progress.

5 Experiments

In this section, we first describe implementation details and the tasks to be completed by the agent. Then, we conduct experiments in multiple tasks from the Mujoco suite [61, 77]. Finally, we answer the following questions:

- What is the impact of the relabeling threshold on the imitator policy training?
- How does the Q-filter method impact the agent's ability to outperform human demonstrations?
- How important is the weight of the imitation loss?
- Is GoAL robust to noisy demonstrations?
- Can GoAL generalize to unseen goals?
- Does our method increase the state coverage of demonstrations?
- Is prioritized goal sampling an efficient way to select goals?
- What is the impact of the query budget on the performance?
- How important is the proposed confidence-based query selection?

5.1 Implementation details and tasks

Experiments are conducted on eight robotic tasks implemented in MujoCo. In the first set of experiments, we consider four manipulation tasks (Fetch tasks) where the agent controls a 7-DoF Sawyer arm: (1) Fetch Reach, (2) Fetch Push, (3) Fetch Pick & Place, and (4) Fetch Slide. The end-effector (EE) is constrained to a 2-dimensional rectangle. In the second set of experiments (ShadowHand tasks), we evaluate our framework on significantly more challenging tasks with very sparse rewards and larger action spaces. The agent is trained to manipulate physical objects via a human-like robot hand: (1) Hand Manipulate Block, (2) Hand Manipulate Egg, (3) Hand Manipulate Pen, and (4) Hand Reach. The observations are given in the form of continuous values and the action-space is also continuous. The performance metric we use is the percentage of goals that the agent is able to reach. An episode is considered successful if the distance between the agent and the goal at the end of the episode is less than a threshold defined by the task.

As our policy learning method, we rely on DDPG with HER. We refer to our algorithm as Goal-driven Active Learning (GoAL). The critic and imitator policy networks consist in 4 fully-connected layers with 256 hidden units. ReLU is used as the activation function expect for the last layer that used tanh, and the output value is scaled to the range of

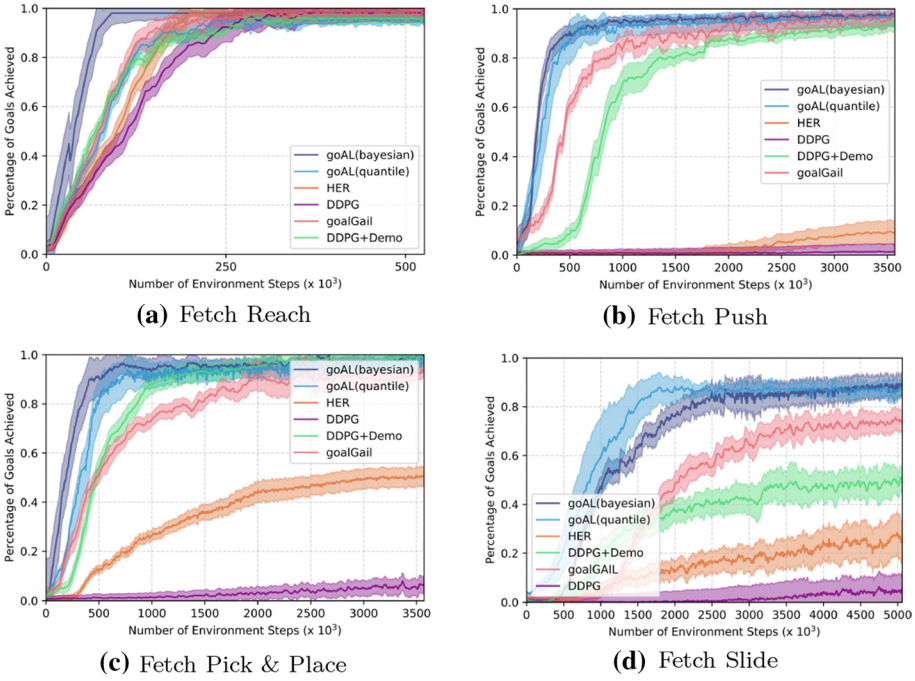


Fig. 3 Learning curves averaged over 10 runs (± SD) for different models: GoAL(bayesian), GoAL(quantile), DDPG, HER, DDPG+Demo, and goalGail. The models are trained on Fetch task

each action dimension. Their parameters are optimized given as input pairs of state-goal. Training is carried out with a fixed learning rate of 10^{-3} using the Adam optimizer [46], with a batch size of 256. Please note that when using quantile-confidence, the last layer of f_ϑ has one output for each quantile. When fitting multiple quantile regressions, it is possible that individual quantile regression estimates overlap, also known as *quantile crossover*

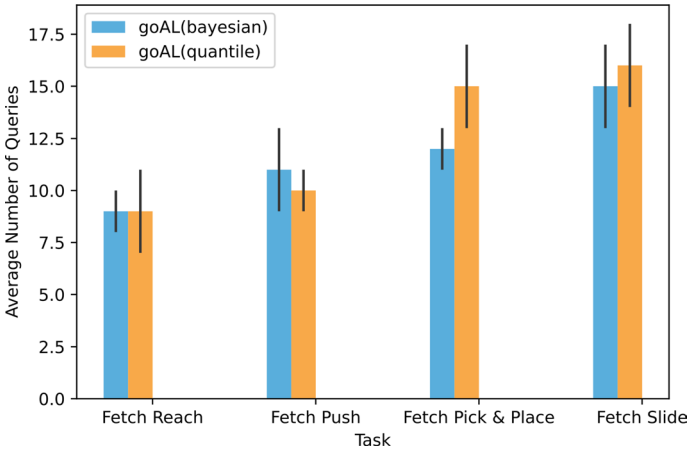


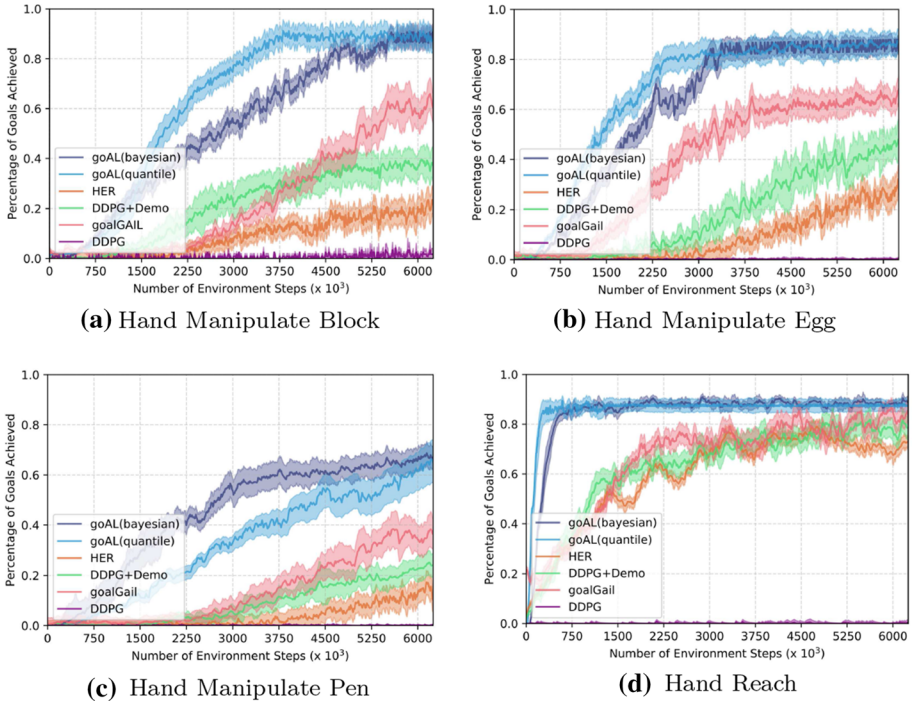
Fig. 4 Average number of queries (± SD) made per environment

[12]—a prediction interval for a lower probability exceeds that of a higher probability. This is unwanted because it does not respect the principle of cumulative distribution functions where their associated inverse functions must be monotonically increasing. In other words, there is no guarantee that the quantile estimates will be rank ordered. This especially happens when fitting the quantiles independently [12, 32]—a multi-headed neural network calculates losses for each desired quantile separately. In order to fit the quantiles better and reduce quantile crossover, we fit the quantiles together at the same time—a single-headed neural network calculates the quantiles together. Note that although this doesn't theoretically guarantee non-crossing regression quantiles, in practice it generally leads the quantiles to be rank ordered.

In order to select the hyperparameters used for Fetch and ShadowHand tasks, we ran a grid search with the ranges shown in Sect. 5.4. We also ran grid searches over the learning rate $\in [0.0001, 0.0005, 0.001, 0.005]$, the number of hidden units $\in [128, 256]$, the query threshold $t_{qry} \in [0.20, 0.21, \ldots, 0.60]$, the number of dropouts $\in [100, 500, 1000]$, and $p \in [0.05, 0.1, 0.2]$. We also searched the annealing factor $\in [0.96, 0.98, 0.99, 0.999]$. As the environments are procedurally generated we performed tuning on the validation set, disjoint with the training set. When tuning, we consider the mean final reward of 10 training runs with the same set of hyperparameters as the objective, without any seed tuning. For Fetch tasks, we use a query budget of 20 and we set $t_{qry} = 0.32$ (Bayesian-confidence) and $t_{qry} = 0.43$ (quantile-confidence). For ShadowHand tasks, we use a query budget of 50, $t_{qry} = 0.58$ (Bayesian-confidence), and $t_{qry} = 0.27$ (quantile-confidence). To generate our demonstrations, we trained DDPG with Hindsight Experience Replay (HER) [4] for 50M environment steps. This allows us to reproduce experiments easily. We used an implementation with the default hyperparameters given in the original work. In all our experiments, Bayesian-confidence is estimated based on 500 dropout masks with $p = 0.1$. The weights of loss components were $\lambda_1 = 1$ and $\lambda_2 = 0.003$ unless stated otherwise and we anneal the imitation loss weight λ_2 by 0.98 per 500 rollouts collected. As relabeling constant N, we set the constant equal to half of the maximum number of time-steps per episode. Since we account for the possibility that the learned policy outperforms expert demonstrations, we employ the depicted Q-filter strategy with $\eta = 0.015$.

5.2 Fetch robotic tasks

We first perform experiments on four different Fetch tasks from the robotic domain built on top of Mujoco: Fetch Reach, Fetch Push, Fetch Pick and Place, and Fetch Slide. We first evaluate DDPG with Hindsight Experience Replay (HER) [4] with and without active-goal driven learning (GoAL). Moreover, we compare our method against several baselines including DDPG [49], DQfD [34], goalGail [19], and DDPG+Demo [54]. Please note that we replace DQN by DDPG as learning algorithm in DQfD. We use 128 demonstrations to guide the baseline methods. To generate these demonstrations, we randomly sample goals and request the expert to reach them. We show learning curves in Fig. 3. Our method can learn comparable or superior policies using a small number of demonstrations. For instance, on Pick and Place, in average only 9 queries were made by GoAL (see Figure 4). As expected, it ends up reaching similar final performance, however, our method has a faster convergence rate. As can be observed, (offline) imitation-based approaches learn fast at the beginning, but their final performance is capped. One reason is that after extracting all task-relevant knowledge from the demonstrations, their convergence speed becomes similar to that of the original HER. On the other hand, incrementally querying



(a) Hand Manipulate Block **(b)** Hand Manipulate Egg

(c) Hand Manipulate Pen **(d)** Hand Reach

Fig. 5 Learning curves (mean ± SD) on ShadowHand tasks averaged over 10 runs for different models: GoAL(bayesian), GoAL(quantile), DDPG, HER, DDPG+Demo, and goalGail

new demonstrations enables us to overcome this problem, while keeping the number of demonstrations very low. Quantile-confidence can be useful to obtain a more comprehensive analysis of the agent's confidence, and is more robust to extreme outliers in the goal-driven demonstrations (e.g. providing the wrong action). Therefore, we believe that this approach should be used when the feedback are provided by a non-expert. On the other hand, Bayesian-confidence is particularly effective to "smooth out" much of the noise in the demonstrations or randomness in the environment, and enables a better generalization of goal-driven demonstrations in the low-data regime. Overall, this experiment highlights that GoAL drastically reduces the training time in sparse and complex environments.

5.3 ShadowHand robotic tasks

In addition to the first robotic tasks, we evaluate our methodology in a more challenging set of environments (ShadowHand): Hand Manipulate Block, Hand Manipulate Egg, Hand Manipulate Pen, and Hand Reach. We provide an expert trajectory dataset of 400 demonstrations to the baseline methods. Figure 5 plots the learning curve of all the models. We can observe that our strategy helps to greatly improve convergence speed. Unlike our algorithm, prior methods passively access the demonstration data, so we actively provide help to our agent when it struggles. On these tasks, we found that goalGAIL does not receive enough supervision to achieve optimal policies. Results highlight that the gap between our approach and the others is increasing with the degree of sparsity. These results further

Table 1 Learning locomotion in Mujoco using different positive thresholds N when training the imitator policy. Results are averaged over 10 random seeds (± SD)

Method	Percentage of goals achieved			
	Fetch Reach	Fetch Push	Fetch Pick & Place	Fetch Slide
GoAL (bayesian)/N = 0.25	0.91 ± 0.03	0.95 ± 0.05	0.90 ± 0.03	0.87 ± 0.04
GoAL (quantile)/N = 0.25	0.90 ± 0.04	0.92 ± 0.02	0.92 ± 0.04	**0.90 ± 0.02**
GoAL (bayesian)/N = 0.50	**0.97 ± 0.02**	0.96 ± 0.04	**0.96 ± 0.04**	0.87 ± 0.06
GoAL (quantile)/N = 0.50	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.03	**0.90 ± 0.05**
GoAL (bayesian)/N = 0.75	0.96 ± 0.03	**0.97 ± 0.05**	0.93 ± 0.05	0.88 ± 0.04
GoAL (quantile)/N = 0.75	0.95 ± 0.04	0.94 ± 0.03	0.93 ± 0.04	0.86 ± 0.04
GoAL (bayesian)/N = 1.0	0.93 ± 0.04	0.94 ± 0.05	0.88 ± 0.05	0.79 ± 0.08
GoAL (quantile)/N = 1.0	0.92 ± 0.05	0.92 ± 0.06	0.91 ± 0.07	0.88 ± 0.04

Bold values indicate the best performing method

No seed tuning is performed

show that since our method is capable of adapting the set of demonstration trajectories to match the learner's goal, GoAL can escape the known "distribution mismatch" issue inherent in standard (offline) imitation learning. Remarkably, the final performance of GoAL is not capped and can even exceed expert-level performance. To the best of our knowledge, this is the first approach operating in the low demonstration regime (less than 50 demonstrations) that achieves a near-optimal score on the four ShadowHand tasks.

5.4 Ablation experiments

We also present an ablation study to investigate: (1) the importance of the relabeling threshold, (2) the impact of the Q-filter, (3) the impact of the weight of the imitation loss, (4) the robustness to imperfect demonstrations, (5) the generalization to unseen goals, (6) the state coverage of queries, (7) the impact of prioritized goal sampling, (8) the size of query budget, and (9) the importance of confidence-based query selection.

5.4.1 Relabeling threshold in imitator policy training

Relabeling the goal-driven demonstrations requires a threshold N to separate "optimal" from "sub-optimal" transitions. Thus, the trained policy implicitly depends on this threshold. Precisely, adding potentially sub-optimal transitions may hurt the performance of the agent. We conduct a study where the threshold N is varied from 0.25 to 1.0. A threshold of 0.25 means that N is equal to one-fourth of the maximum of time-steps per episode. Although all the goals can be used for relabeling, we find this solution less than ideal (see Table 1) in some cases and may even hurt the performance. This can happen for mainly two reasons: (1) conflicts arising from sub-optimal samples recorded in the demonstration buffer, and (2) since the agent's confidence for imaginary sub-optimal samples is high, the agent struggles to identify regions where feedback is the most needed. On the other hand, the results show that a reasonable choice of N is between 0.5 or 0.75.

5.4.2 Importance of the Q-filter

One of the promises of our approach is its potential ability to exceeded the expert performance. In this experiment, we verify if this promise holds. To this end, we conduct a study where the threshold η of the Q-filter is varied from 0.001 to 0.1 as well as the method training without Q-filter. Table 2 reports results of our methods, GoAL, on the four Fetch tasks with different η. We can observe that using Q-filter allows the agent to outperform the teacher by discarding sub-optimal demonstrations. On the other hand, the performance of the agent trained without Q-filter are capped and cannot exceed the expert performance. Please note that when η is too large (e.g. $\eta = 0.1$), the agent does not take into account the demonstrations that are slightly better than its policy, slightly reducing its performance. Table 2 further shows that the parameter η does not require to be fine-tuned for each task (i.e. $0.03 \geq \eta \geq 0015$) since the agents maintain acceptable performance. It is observed that for most tasks, setting $\eta = 0.03$ or $\eta = 0.0015$ produced an effective trade-off between exploiting the expert's knowledge and leveraging the agent's knowledge about the task.

5.4.3 Weight of imitation loss

Combining the policy loss and imitation loss involves a hyperparameter λ_2, which weights the importance of the imitation loss. This brings up an interesting question—what is the impact of this hyperparameter on the performance of the agent? Ideally, the policy performance should not be too sensitive to this hyperparameter. We perform a study for various values of λ_2 in 0.001, 0.003, 0.006. Table 3 shows that the GoAL performance is robust to the choice of this hyperparameter.

Table 2 Learning locomotion in Mujoco using different Q-filter thresholds η

Method	Percentage of goals achieved			
	Fetch Reach	Fetch Push	Fetch Pick & Place	Fetch Slide
GoAL (bayesian)/$\eta = 0.1$	0.92 ± 0.02	0.91 ± 0.03	0.93 ± 0.03	0.87 ± 0.04
GoAL (quantile)/$\eta = 0.1$	0.95 ± 0.03	0.90 ± 0.04	0.92 ± 0.05	0.64 ± 0.05
GoAL (bayesian)/$\eta = 0.03$	0.96 ± 0.03	0.94 ± 0.02	$\mathbf{0.97 \pm 0.04}$	0.86 ± 0.05
GoAL (quantile)/$\eta = 0.03$	$\mathbf{0.97 \pm 0.04}$	0.92 ± 0.03	0.95 ± 0.05	0.85 ± 0.03
GoAL(bayesian)/$\eta = 0.015$	$\mathbf{0.97 \pm 0.02}$	$\mathbf{0.96 \pm 0.04}$	0.96 ± 0.04	0.87 ± 0.06
GoAL (quantile)/$\eta = 0.015$	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.03	$\mathbf{0.90 \pm 0.05}$
GoAL (bayesian)/$\eta = 0.001$	0.89 ± 0.03	0.88 ± 0.05	0.90 ± 0.03	0.77 ± 0.05
GoAL (quantile)/$\eta = 0.001$	0.87 ± 0.02	0.89 ± 0.04	0.85 ± 0.06	0.73 ± 0.04
GoAL (bayesian)/No Q-filter	0.75 ± 0.05	0.71 ± 0.06	0.77 ± 0.12	0.68 ± 0.08
GoAL (quantile)/No Q-filter	0.69 ± 0.08	0.72 ± 0.09	0.70 ± 0.07	0.64 ± 0.09

Bold values indicate the best performing method

Results are averaged over 10 random seeds ($\pm$ SD). No seed tuning is performed

Table 3 Learning locomotion in Mujoco using different imitation loss weights

Method	Percentage of goals achieved			
	Fetch Reach	Fetch Push	Fetch Pick & Place	Fetch Slide
GoAL (bayesian) / $\lambda_2 = 0.001$	0.95 ± 0.03	0.93 ± 0.05	0.94 ± 0.07	0.88 ± 0.08
GoAL (quantile) / $\lambda_2 = 0.001$	0.96 ± 0.04	0.92 ± 0.04	0.91 ± 0.05	0.89 ± 0.05
GoAL (bayesian) / $\lambda_2 = 0.003$	$\mathbf{0.97 \pm 0.02}$	$\mathbf{0.96 \pm 0.04}$	$\mathbf{0.96 \pm 0.04}$	0.87 ± 0.06
GoAL (quantile) / $\lambda_2 = 0.003$	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.03	$\mathbf{0.90 \pm 0.05}$
GoAL (bayesian) / $\lambda_2 = 0.006$	$\mathbf{0.97 \pm 0.02}$	0.95 ± 0.06	0.92 ± 0.05	$\mathbf{0.90 \pm 0.04}$
GoAL (quantile) / $\lambda_2 = 0.006$	0.94 ± 0.04	0.93 ± 0.03	0.93 ± 0.04	0.86 ± 0.06

Bold values indicate the best performing method

Results are averaged over 10 random seeds ($\pm$ SD). No seed tuning is performed

5.4.4 Robustness to imperfect demonstrations

In the above experiments, we assume perfect demonstrations. However, the expert might select not the best action or even lack knowledge about a goal. We study how our agents perform when imperfect demonstrations are generated by the demonstrators. In order to generate imperfect demonstrations, we add normal noise $\mathcal{N}(0, \sigma^2)$ to the teacher actions with a probability $\varrho \in \{0.05, 0.1, 0.2\}$ and $\sigma = 0.03$. We report in Table 4 the performance of our framework and several baselines. We observe that GoAL can still achieve acceptable performance. For instance, the success rate of the proposed method remains larger than 0.80 on the three tasks ($\varrho = 0.05$). Even though GoAL(bayesian) performs slightly worse in the imperfect setting, it still improves performance as compared to the prior methods. A reason is that dropout allows the imitator to "smooth out" much of the noise in the data, making GoAL(bayesian) robust to noisy demonstrations. Moreover, annealing the imitation loss weight and filtering the sub-optimal demonstrations allow us to escape from poor local optima, improving significantly beyond the (imperfect) expert demonstrations. The results demonstrate that our method is reasonably robust to noise in the demonstrations, and hence non-expert can provide a feedback signal to the agent.

5.4.5 Generalization to unseen goals

In the previous section, we showed that our method learns to achieve a wide range of goals. However, it remains unclear whether the agent has achieved this by "generalizing demonstrated trajectories". To investigate this question, we train our agent on a set of goals and evaluate its performance on a different set of goals (without additional queries). From Table 5, we see that the agent can generalize to unseen goals, with a slight loss in the performance. As the agent has already learned about parts of the environment, it can leverage known similar goals to face an unseen situation. We can further observe that GoAL(bayesian) tends to generalize to unseen situations better than GoAL(quantile). We hypothesize that using dropout in the imitator policy network prevents the network to overfit the provided goal-driven demonstrations. Moreover, the results highlight that Bayesian-confidence is slightly more accurate to estimate the agent's confidence than quantile-confidence, improving the value information of the queries. Experimental results suggest that

Table 4 Percentage of goals achieved from imperfect demonstrations, where ϱ is the probability of providing a sub-optimal action

	ϱ=0.05			ϱ=0.1			ϱ=0.2		
Method	Fetch Push	Fetch Pick & Place	Fetch Slide	Fetch Push	Fetch Pick & Place	Fetch Slide	Fetch Push	Fetch Pick & Place	Fetch Slide
goalGail	0.91 ± 0.04	0.84 ± 0.06	0.72 ± 0.08	0.88 ± 0.05	0.78 ± 0.11	0.70 ± 0.09	0.74 ± 0.08	0.62 ± 0.14	0.62 ± 0.13
DDPG + Demo	0.89 ± 0.05	0.81 ± 0.07	0.48 ± 0.08	0.72 ± 0.08	0.62 ± 0.14	0.30 ± 0.12	0.51 ± 0.15	0.47 ± 0.18	0.21 ± 0.20
GoA L(bayesian)	**0.94 ± 0.06**	0.83 ± 0.06	0.90 ± 0.10	**0.91 ± 0.08**	**0.80 ± 0.10**	**0.87 ± 0.12**	**0.85 ± 0.06**	**0.77 ± 0.08**	**0.84 ± 0.14**
GoAL (quantile)	0.92 ± 0.05	**0.94 ± 0.13**	**0.91 ± 0.07**	0.88 ± 0.08	0.72 ± 0.14	0.69 ± 0.07	0.77 ± 0.13	0.66 ± 0.12	0.62 ± 0.10

Bold values indicate the best performing method

The results are averaged across 10 random seeds (± SD). We set the number of queries to 20

Table 5 Evaluation of the agent trained on several Mujoco tasks ("with fine-tuning")

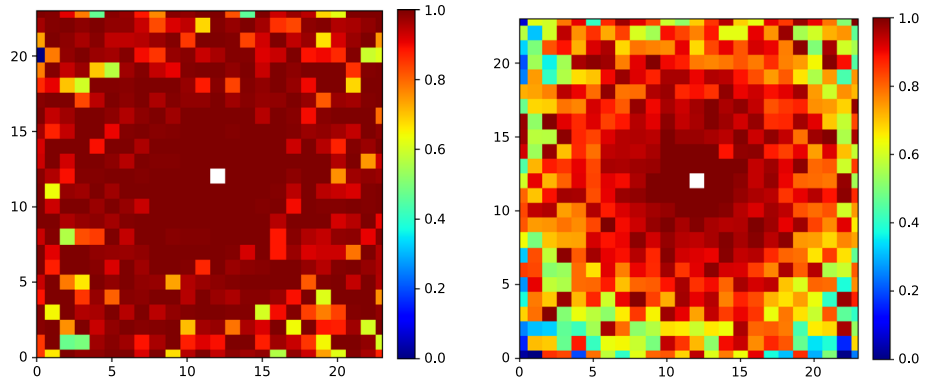
Method	Percentage of goals achieved			
	Fetch Reach (0.5M)	Fetch Push (3M)	Fetch Pick & Place (3M)	Fetch Slide (3M)
With fine-tuning (bayesian)	0.97 ± 0.02	0.96 ± 0.04	0.96 ± 0.04	0.87 ± 0.06
With fine-tuning (quantile)	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.03	0.90 ± 0.05
Without fine-tuning (bayesian)	0.95 ± 0.03	0.84 ± 0.08	0.93 ± 0.11	0.85 ± 0.10
Without fine-tuning (quantile)	0.93 ± 0.04	0.79 ± 0.06	0.85 ± 0.08	0.72 ± 0.09

To investigate whether the agent is able to generalize the demonstrated trajectories, we then evaluate the agent's performance on a different set of goals without querying new demonstrations ("without fine-tuning"). We report the results averaged over 10 seeds ($\pm$ SD)

the set of collected trajectories provides a wide enough state coverage, leading to an efficient generalization.

5.4.6 State coverage of queries

In this experiment, we show that our method is efficient at querying demonstrations and can keep the level of redundancy at a minimum. To do so, we show state visitation heatmaps of trajectories queried by our method over 100 runs on a simple 2D navigation environment. The agent starts in the center of the box, and can take actions to directly move its position. In this task, the agent needs to navigate itself to a target position (x,y) that is randomly generated by the environment. Figure 6 illustrates that trajectories requested using our method cover most of the states. One reason is that similar goals are not requested for demonstrations since the confidence is large enough. Thus, the agent explores further and makes queries in places where it struggles. On the other hand, as the agent (without



(a) With confidence **(b)** Without confidence

Fig. 6 State visitation heatmaps of the demonstrations queried by the GoAL agent with Bayesian-confidence (left) and without confidence prediction (right). The agent starts in the middle of the board (white square) and has to navigate to a target position chosen by the environment. The left figure shows that the proposed method to select queries significantly increases state coverage compared to the agent trained without confidence-bases query selection (right)

Table 6 Learning locomotion in Mujoco, with and without prioritized goal sampling (PGS)

	Percentage of goals achieved (convergence speed $\times 10^3$ training steps)			
Method	Fetch Reach	Fetch Push	Fetch Pick & Place	Fetch Slide
GoAL (bayesian)	**0.97 (122)**	**0.96 (982)**	**0.96 (812)**	0.87 (1850)
GoAL (quantile)	0.96 (241)	0.94 (1023)	0.95 (1073)	**0.90 (2609)**
GoAL (bayesian) without PGS	0.96 (157)	0.97 (1256)	0.95 (1229)	0.85(2044)
GoAL (quantile) without PGS	0.96 (270)	0.95 (883)	0.94 (1627)	**0.91 (2953)**

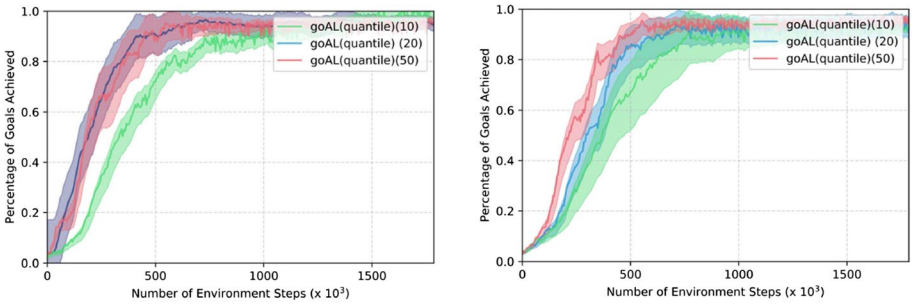
Bold values indicate the best performing method

Results are averager over 10 random seeds. No seed tuning is performed

Bayesian-confidence) fails to reach the goals being pursued, it quickly exhausts its query budget by making redundant queries. As a result, we can expect imitation learning to be efficient only near the center of the box. This issue becomes even more noticeable as the size or the complexity of the environment is increased. In general we discovered that the depicted algorithm improves state coverage and eliminates the need for unnecessary demonstrations of already acquired behaviors.

5.4.7 Using prioritized goal sampling

One legitimate question is to study the impact of the prioritized goal sampling on the performance of the algorithm. We conduct a study with and without prioritized goal sampling (PGS). As shown in Table 6, PGS produces faster convergence speed in three environments, including Fetch Reach, Fetch Pick & Place, and Fetch Slide. For example, on Fetch Reach, PGS reduces the number of interactions to converge by $\approx 28\%$ for GoAL(bayesian) and $\approx 12\%$ for GoAL(quantile). Furthermore, the results show that PGS does not significantly deteriorate performance, and in some cases our agent can reach higher final performance than running pure goal sampling. Please note that PGS slightly deteriorates performance on both Fetch Push and Fetch Slide but also generally reduces the number of necessary training steps. Overall, it confirms that the most important goals for replays are the ones where the disagreement in the prediction is maximized.



(a) GoAL(bayesian) **(b)** GoAL(quantile)

Fig. 7 Learning curves averaged over 10 runs (± SD) on the Fetch Pick & Place task, with different query budgets

5.4.8 Query budget

We also report evaluations showing the effect of increased query budget. Figure 7 demonstrates that agents trained with a larger query budget obtain higher mean returns after similar numbers of updates. However, despite a small query budget, our method can still learn near optimal policies. We can draw the observation that as the query budget increases, the learning effect on the agent gradually improves. However, for the results with 20 and 50 queries, we can see that even if the number of queries significantly differs, the difference in learning effect can be negligible. This can happen when queried demonstrations cover a broad state space and therefore the agent does not need to make additional queries. As a result, our method leverages a small amount of demonstrations that cover task-relevant regions of the state space and outperforms the baselines by a large margin (see Sect. 5.2).

5.4.9 Importance of confidence-based query selection

Finally, to quantify the importance of confidence-based query selection, we compare the performance of our architecture with and without confidence-based query selection. Table 7 plots the mean episode-returns obtained for different tasks. We observe that including confidence-based query selection always leads to notably better episode-returns, indicating that it is useful to select queries based on the agent's confidence. We observed that using confidence-based query selection helps the agent to select task-relevant queries that will have a large impact on its learning progress. On the other hand, the agents trained without confidence check quickly exhaust their query budget by querying goals easy to reach or goals similar to already encountered situations, leading to an abundance of irrelevant or unnecessary demonstrations.

6 Discussion

We have constructed a mechanism to utilize goal-driven demonstrations along with goal-conditioned reinforcement learning. In order to greatly reduce the number of required demonstrations, we propose to query the demonstrator in states where the agent struggles and the confidence in the action prediction is low. This concern is relevant when cooperating with human teachers, in particular since human effort is limited by factors such as attention span or cost of interactions [18]. Thus, it is desirable to limit the number of queries to only those that are most needed. In the proposed method, even

Table 7 Learning locomotion in Mujoco with and without confidence-based query selection

Method	Percentage of goals achieved			
	Fetch Reach	Fetch Push	Fetch Pick & Place	Fetch Slide
GoAL (bayesian)	**0.97 ± 0.02**	**0.96 ± 0.04**	**0.96 ± 0.04**	0.87 ± 0.06
GoAL (quantile)	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.03	**0.90 ± 0.05**
GoAL (no confidence)	0.82 ± 0.07	0.61 ± 0.03	0.65 ± 0.03	0.53 ± 0.06

Bold values indicate the best performing method

Results are averaged over 10 random seeds (± SD). No seed tuning is performed

very small amounts of queries (less than 50 queries) let us outperform prior imitation-based approaches on Fetch and ShadowHand tasks. These games involve sparse and delayed rewards. These results suggest that GoAL can greatly benefit in exploration efficiency and could help to expand the possible applications of RL.

That being said, we acknowledge that our approach has certain limitations. HER relies on a reward function $R(s, a, g)$ to relabel additional goals used for replay. Deriving a good reward function is not straightforward in environments where goals are images. One solution inspired by RIG [55] is to measure the distance between two images in a latent space. We leave it to future work to explore this direction further.

Another limitation is the need for demonstrations which can be challenging in some environments. If demonstrations are not available, one solution is to reuse successful rollouts as demonstration data [27]. It may also be possible to reuse imperfect agent policies trained on similar tasks to generate the demonstrations.

So far, ablation analysis in Sect. 5.4 confirmed the above-mentioned intuitions but lacked theoretical understanding. In future work, we aim to improve the theoretical understanding of our approach, more specifically, how Q-filter relates to the capability of the algorithm for outperforming expert demonstrators, convergence speed, and how the number of queries contributes to GoAL performance. In standard passive imitation learning, several work such as [82] or [13], have been dedicated to provide theoretical and safety guarantees. In active imitation learning, similar attempts [42, 43] were made. Despite the significant empirical progresses, many theoretical aspects remain largely unknown. The major difficulty comes from the underlying temporal dependency of the demonstration data and the difficulty to provide guarantees due to human factors. Nevertheless, developing strong theoretical guarantees will be useful to deploy our system in the real world.

In the current version of GoAL, the imitation relies on a mean squared error. A potentially more efficient imitation approach would be to use techniques related to Batch RL [15, 29]. In detail, it was shown that cloning a narrow set of expert trajectories using standard off-policy RL can result in extrapolation error [30]. Namely, the agent quickly learns to select non-expert actions under the guise of optimistic extrapolation. However, the agent does not consider the accuracy of the estimate. As a result, unfamiliar state-action transitions outside the batch may be viewed over-optimistically. A solution to address the extrapolation issue is to add a batch constraint to the off-policy algorithm [30], which we leave for future work.

One promising direction is to replace the human expert with another agent's advice. After the learner determines when to query a prospective teacher, it may be possible to query another agent that has already acquired knowledge about the situation. This inter-agent teaching strategy has been used to solve tasks such as video game playing [76], but it remains an open problem in complex tasks [18]. Another possible solution could be to train the agent on a mixture of demonstration data collected from an agent and a human. We believe that exploring multi-agent collaboration is an important direction in order to further reduce human effort.

Another exciting future direction is to train GoAL on physical robots. It has been shown that HER trained on simulated data can be deployed on a physical fetch robot [4]. However, directly training DRL on physical robots remains an open problem. We can expect our method to benefit in sample efficiency and to significantly reduce the number of interactions.

7 Conclusion

In this paper we presented Goal-driven Active Learning (GoAL), a method introducing interactive goal-driven demonstrations to both learn more effectively and efficiently. Goal-driven demonstrations do not intend to demonstrate the overall task, but help the agent to fulfill particular intermediate goals when it struggles. This novel form of human guidance is less expensive and more intuitive than pure demonstrations, while ensuring that the provided knowledge match the agent's needs, hence escaping the known "distribution mismatch" issues of prior work. Unlike traditional methods of imitation learning where the agent passively accesses to the demonstration data, our method actively decides when to request demonstrations based on the confidence of the agent. We introduced and studied the effect of two strategies to measure the model's confidence. Finally, we proposed to promote goal sampling in places where the agent strongly disagrees with the demonstrator, and we experimentally showed that it improves the performance of GoAL. Our method shows substantial improvements over prior work in the Mujoco benchmark suite. Remarkably, the depicted algorithmic framework can match the basic demonstration-level performance and even exceed expert-level performance. Furthermore, GoAL learns to reach a wide range of configurations from the same set of demonstrated trajectories. These results suggest that GoAL can greatly benefit in exploration efficiency and could help to expand the possible applications of RL. In the long run it would be desirable to incorporate adaptive confidence threshold to further improve query selection. Another intriguing direction for future work is to drive learning in new tasks in a few-shot style.

References

1. Abbeel, P., & Ng, A. Y. (2004). *Apprenticeship learning via inverse reinforcement learning* (pp. 1–8).
2. Agarwal, P., de Beaucorps, P., & de Charette, R. (2021). Sparse curriculum reinforcement learning for end-to-end driving. 2103.09189.
3. Amir, O., Kamar, E., Kolobov, A., & Grosz, B. J. (2016). Interactive teaching strategies for agent training. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 804–811).
4. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O.P., & Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems* (pp. 5048–5058).
5. Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems, 57*(5), 469–483.
6. Bassich, A., Foglino, F., Leonetti, M., & Kudenko, D. (2020). Curriculum learning with a progression function. arXiv preprint arXiv:200800511.
7. Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Proceedings of advances in neural information processing systems* (pp. 1471–1479).

8. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).
9. Bougie, N., & Ichise, R. (2019). Skill-based curiosity for intrinsically motivated reinforcement learning. *Machine Learning,* 493–512.
10. Bougie, N., Cheng, L. K., & Ichise, R. (2018). Combining deep reinforcement learning with prior knowledge and reasoning. *ACM SIGAPP Applied Computing Review, 18*(2), 33–45.
11. Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2019). Exploration by random network distillation. In *Proceedings of the international conference on learning representations*.
12. Cannon, A. J. (2018). Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. *Stochastic Environmental Research and Risk Assessment, 32*(11), 3207–3225. https://doi.org/10.1007/s00477-018-1573-6
13. Chen, M., Wang, Y., Liu, T., Yang, Z., Li, X., Wang, Z., & Zhao, T. (2020). On computation and generalization of generative adversarial imitation learning. In *International conference on learning representations*
14. Chen, S. A., Tangkaratt, V., Lin, H. T., & Sugiyama, M. (2019). Active deep q-learning with demonstration. *Machine Learning*, 1–27.
15. Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., & Ross, K. (2020b). Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems* 33
16. Chernova, S., & Veloso, M. (2007). Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the international joint conference on autonomous agents and multiagent systems* (pp. 1–8).
17. Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural information processing systems* (pp. 4299–4307).
18. Da Silva, F. L., Warnell, G., Costa, A. H. R., & Stone, P. (2020). Agents teaching agents: Asurvey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems, 34*(1), 1–17.
19. Ding, Y., Florensa, C., Abbeel, P., & Phielipp, M. (2019). Goal-conditioned imitation learning. In *Advances in neural information processing systems* (pp. 15298–15309).
20. Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., Abbeel, P., & Zaremba, W. (2017). One-shot imitation learning. In *Advances in neural information processing systems* (Vol. 30).
21. Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2019). Diversity is all you need: Learning skills without a reward function. In *International conference on learning representations*.
22. Fachantidis, A., Taylor, M. E., & Vlahavas, I. (2019). Learning to teach reinforcement learning agents. *Machine Learning and Knowledge Extraction, 1*(1), 21–42.
23. Fan, Y., Tian, F., Qin, T., Li, X. Y., & Liu, T. Y. (2018). Learning to teach. In *International conference on learning representations*.
24. Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135).
25. Finn, C., Yu, T., Zhang, T., Abbeel, P., & Levine, S. (2017). One-shot visual imitation learning via meta-learning. In *Proceedings of the 1st annual conference on robot learning, Proceedings of machine learning research* (Vol. 78, pp. 357–368).
26. Florensa, C., Held, D., Geng, X., & Abbeel, P. (2017). Automatic goal generation for reinforcement learning agents. arXiv preprint: arXiv:170506366.
27. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017) Reverse curriculum generation for reinforcement learning. arXiv preprint arXiv:170705300.
28. Forestier, S., Mollard, Y., & Oudeyer, P.Y. (2017). Intrinsically motivated goal exploration processes with automatic curriculum learning. arXiv preprint: arXiv:170802190.
29. Fujimoto, S., Conti, E., Ghavamzadeh, M., & Pineau, J. (2019). Benchmarking batch deep reinforcement learning algorithms. arXiv preprint arXiv:191001708
30. Fujimoto, S., Meger, D., & Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International conference on machine learning, PMLR* (pp. 2052–2062).
31. Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the international conference on machine learning* (pp. 1050–1059).
32. Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., & Januschowski, T. (2019). Probabilistic forecasting with spline quantile function rnns. In *The 22nd international conference on artificial intelligence and statistics* (pp. 1901–1910).

33. Graves, A., Bellemare, M. G., Menick, J., Munos, R., & Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In *Proceedings of the 34th international conference on machine learning* (pp. 1311–1320).
34. Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J. Z., & Gruslys, A. (2018). Deep q-learning from demonstrations. In *Proceedings of the annual meeting of the association for the advancement of artificial intelligence*.
35. Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems* (pp. 4565–4573).
36. Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Proceedings of advances in neural information processing systems* (pp 1109–1117).
37. Hsu, D. (2019). A new framework for query efficient active imitation learning. arXiv preprint arXiv:191213037.
38. Huang, S., & Ontañón, S. (2020). Action guidance: Getting the best of sparse rewards and shaped rewards for real-time strategy games. 2010.03956.
39. Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., & Amodei, D. (2018). Reward learning from human preferences and demonstrations in atari. In *Advances in neural information processing systems* (pp 8011–8023).
40. James, S., Bloesch, M., & Davison, A. J. (2018). Task-embedded control networks for few-shot imitation learning. In *Conference on robot learning* (pp 783–795).
41. John, O. O. (2015). Robustness of quantile regression to outliers. *American Journal of Applied Mathematics and Statistics, 3*(2), 86–88.
42. Judah, K., Fern, A., & Dietterich, T.G. (2012). Active imitation learning via reduction to iid active learning. arXiv preprint arXiv:12104876.
43. Judah, K., Fern, A. P., Dietterich, T. G., & Tadepalli, P. (2014). Active imitation learning: Formal and practical reductions to i.i.d. learning. *Journal of Machine Learning Research, 15*(120), 4105–4143.
44. Kaelbling, L. P. (1993). Learning to achieve goals. In *Proceedings of the international joint conferences on artificial intelligence* (pp 1094–1098).
45. Kang, B., Jie, Z., & Feng, J. (2018). Policy optimization with demonstrations. In *International conference on machine learning* (pp. 2469–2478).
46. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980.
47. Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. *Proceedings of the IEEE Congress on Evolutionary Computation, 1*, 128–135.
48. Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research, 17*(1), 1334–1373.
49. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, . (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971.
50. Machado, M. C., Bellemare, M. G., & Bowling, M. (2018). Count-based exploration with the successor representation. arXiv preprint arXiv:180711622
51. Martin, J., Sasikumar, S. N., Everitt, T., & Hutter, M. (2017). Count-based exploration in feature space for reinforcement learning. In *Proceedings of the international joint conference on artificial intelligence*.
52. Mathewson, K.W., Pilarski, P.M. (2017). Actor-critic reinforcement learning with simultaneous human control and feedback. arXiv preprint arXiv:170301274.
53. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., & Georg. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
54. Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Overcoming exploration in reinforcement learning with demonstrations. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 6292–6299). IEEE.
55. Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., & Levine, S. (2018b). Visual reinforcement learning with imagined goals. In *Proceedings of the international conference on machine learning* (pp. 9191–9200).
56. Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the international conference on machine learning* (pp 663–670).
57. Ostrovski, G., Bellemare, M. G., van den Oord, A., & Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the international conference on machine learning* (pp. 2721–2730)

58. Oudeyer, P. Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation, 11*(2), 265–286.
59. Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the international conference on international conference on machine learning* (pp. 2778–2787).
60. Pere, A., Forestier, S., Sigaud, O., & Oudeyer, P. Y. (2018). Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In *Proceedings of the international conference on learning representations*.
61. Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., & Zaremba, W. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. arXiv preprint arXiv:180209464.
62. Pomerleau, D. A. (1988). Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 1st international conference on neural information processing systems* (pp 305–313).
63. Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., & Levine, S. (2019). Skew-fit: State-covering self-supervised reinforcement learning. arXiv preprint:190303698.
64. Racaniere, S., Lampinen, A., Santoro, A., Reichert, D., Firoiu, V., & Lillicrap, T. (2020). Automated curriculum generation through setter-solver interactions. In *International conference on learning representations*
65. Saunders, W., Sastry, G., Stuhlmueller, A., & Evans, O. (2018). Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the international conference on autonomous agents and multiAgent systems* (pp. 2067–2069).
66. Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., & Gelly, S. (2019). Episodic curiosity through reachability. In *Proceedings of the international conference on learning representations*.
67. Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences, 3*(6), 233–242.
68. Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *Proceedings of the international conference on machine learning* (pp 1312–1320).
69. Shon, A. P., Verma, D., & Rao, R. P. (2007). Active imitation learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 756–762).
70. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484.
71. Stadie, B. C., Levine, .S, & Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. arXiv preprint arXiv:150700814.
72. Strehl, A. L., & Littman, M. L. (2008). An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences, 74*(8), 1309–1331.
73. Sun, W., Venkatraman, A., Gordon, G.J., Boots, B., Bagnell, J.A. (2017). Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 3309–3318).
74. Tagasovska, N., & Lopez-Paz, D. (2019). Single-model uncertainties for deep learning. *Advances in Neural Information Processing Systems, 32*, 6417–6428.
75. Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2017). # exploration: Astudy of count-based exploration for deep reinforcement learning. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 2750–2759).
76. Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I., & Torrey, L. (2014). Reinforcement learning agents providing advice in complex video games. *Connection Science, 26*(1), 45–63.
77. Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033).
78. Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., & Riedmiller, M. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:170708817.
79. Warnell, G., Waytowich, N., Lawhern, V., & Stone, P. (2018). Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Thirty-second AAAI conference on artificial intelligence* (pp 1545–1554).
80. Wilson, A., Fern, A., Tadepalli, P. (2012). A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems* (pp. 1133–1141).

81. Wirth, C., Akrour, R., Neumann, G., & Fürnkranz, J. (2017). A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research, 18*(1), 4945–4990.
82. Yin, H., Seiler, P., Jin, M., & Arcak, M. (2020). Imitation learning with stability and safety guarantees. arXiv preprint arXiv:201209293.
83. Zaremba, W., & Sutskever, I. (2015). Learning to execute. 1410.4615.
84. Zhang, X., & Ma, H. (2018). Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations. arXiv preprint arXiv:180110459.
85. Zimmer, M., Viappiani, P., & Weng, P. (2014). Teacher-student framework: Areinforcement learning approach. In *AAMAS workshop autonomous robots and multirobot systems*.
86. Zuo, G., Zhao, Q., Lu, J., & Li, J. (2020). Efficient hindsight reinforcement learning using demonstrations for robotic tasks with sparse rewards. *International Journal of Advanced Robotic Systems, 17*.