WayEx: Waypoint Exploration using a Single Demonstration

Mara Levy, Nirat Saini, and Abhinav Shrivastava

University of Maryland, College Park

***Abstract*— We propose WayEx, a new method for learning complex goal-conditioned robotics tasks from a single demonstration. Our approach distinguishes itself from existing imitation learning methods by demanding fewer expert examples and eliminating the need for information about the actions taken during the demonstration. This is accomplished by introducing a new reward function and employing a knowledge expansion technique. We demonstrate the effectiveness of WayEx, our waypoint exploration strategy, across six diverse tasks, showcasing its applicability in various environments. Notably, our method significantly reduces training time by ∼50% as compared to traditional reinforcement learning methods. WayEx obtains a higher reward than existing imitation learning methods given only a single demonstration. Furthermore, we demonstrate its success in tackling complex environments where standard approaches fall short. Appendix is available at: https://waypoint-ex.github.io.**

I. INTRODUCTION

Humans have a natural ability to learn tasks by observing a single demonstration which they can follow step-by-step. For instance, we can watch a video and grasp how to open a vault, then practice until we succeed without requiring further instructions. Drawing inspiration from this ability, the combination of learning from demonstrations and reinforcement learning techniques has become a popular and potent approach for training robots [1, 2, 3, 4]. However, compared to humans who can learn simple tasks from a single demonstration, robots require a multitude of diverse expert instances. For example, to learn how to open a vault, the robot must observe successful demonstrations for different views and locations of the vault handle with respect to the robot's location. Moreover, each demonstration must contain information about the location of the vault (state), the precise joint rotations (action) to reach the vault, and knowledge about how close it is to completing the task (reward). Hence, most common methods in learning from demonstrations, such as Imitation learning [5, 6, 4] and Inverse reinforcement learning [7, 8, 9] require a set of expert demonstrations, with a defined state, action and reward space. Collecting all the data in real time and computing the definitive action and reward space is impractical and inefficient. Therefore, in this work, we strive to reduce these inefficiencies by using only a single demonstration for training. Additionally, our setup does not require knowledge about the action space, relying on just the state space and a corresponding reward function.

Prior works identify key states (waypoints [6, 5]) along the robot's trajectory, to help it navigate towards the goal. We also employ waypoints to solve the task, without having access to the action space. Each observation within the demonstration is defined as an individual waypoint. While other approaches [6] need access to waypoints during the testing phase, WayEx only requires waypoints during training. This prevents the need for training a model to predict the waypoints during inference. We leverage the waypoints during training via an augmented reward structure based on the known Q-Values [10] associated with each waypoint.

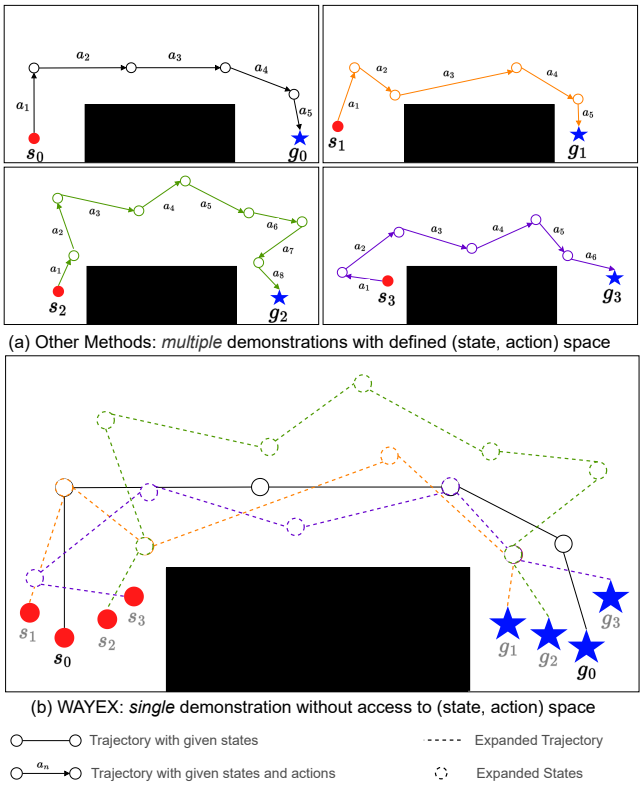


Fig. 1. A comparison of our approach to general imitation learning techniques. **(a)** Traditional Imitation learning approaches require multiple expert trajectories with a known action space for training (4 shown here). **(b)** For our proposed method WayEx we use only one expert trajectory, and expand knowledge from this one trajectory to learn how to solve the task. During training with a single initial state (s_0) and a single goal state (g_0), our model learns to navigate back to the expert trajectory from points that are not part of the trajectory (all dotted states, which can be a combination of 4 expert trajectories shown on the top). This enables the model to successfully reach the goal state. We further introduce additional start and goal states ($[s_1, g_1]$,[s_2, g_2], $[s_3, g_3]$).

A common approach to solving a task without a dataset of expert trajectories is to use dense rewards, based on the key steps of a task. Existing studies in the field of reinforcement learning acknowledge that employing dense rewards is difficult since it requires practitioners to engineer specific reward functions for each task. Additionally these rewards, if ill-designed, can lead to unforeseen behavior. To circumvent these challenges, we assume a sparse reward structure when

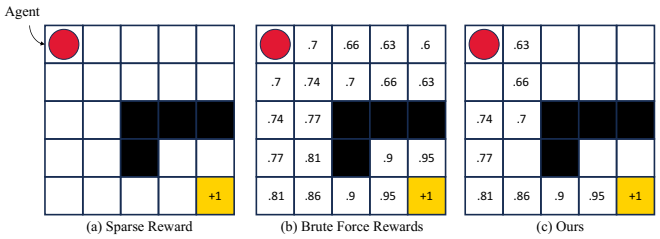
determining the new reward. With sparse rewards, the robot receives a reward of 0 if the actions lead to a goal state; otherwise, the reward is -1. Acquiring this reward solely through the process of exploration can prove to be highly challenging, making certain tasks unachievable with sparse rewards alone [11]. Our approach strikes a balance, allowing the model to receive frequent rewards without exposing it to the typical risks associated with dense rewards.

By utilizing this new reward structure, our model can solve tasks that closely resemble the demonstration setup. However, despite its ability to learn from a single example, our approach still encounters a common limitation of learning from demonstrations. If the robot encounters a state beyond the scope of what it has previously encountered it will not know how to proceed. To overcome this challenge, the robot needs to acquire experience beyond the confines of the provided demonstrated space. A commonly employed method to achieve this is to integrate learning from demonstrations with conventional model-free reinforcement learning algorithms [12, 13]. Strict model-free reinforcement learning involves learning optimal state-action pairs through trial and error rather than relying on expert trajectories. We combine model-free reinforcement learning with our waypoint reward and introduce an additional expansion method that further enhances the model's knowledge.

In this work, we propose WayEx, derived from **Way**point **Ex**ploration, a novel approach that enables the training of a reinforcement learning model using a single expert demonstration and without any prior knowledge of the action space. It can serve as a wrapper around any reinforcement learning algorithm, facilitating its applicability as the field advances. Our primary technical contributions are: (1) the introduction of a new reward function based on sparse rewards, which provides additional rewards without introducing unforeseen consequences, and (2) a method for expanding knowledge beyond a single demonstration to encompass the entire spectrum of both the state and goal spaces. We demonstrate that our approach enables faster learning of tasks compared to previous reinforcement learning methods while requiring minimal additional information.

II. RELATED WORK

Goal-Conditioned Reinforcement Learning. We are interested in investigating tasks that involve a robot reaching a specific end state specified by an initial "goal." In the reinforcement learning (RL) community, these problems are known as Goal-Conditioned Tasks. Prior works have studied how to use RL in many different ways in order to solve these tasks [14, 15, 16, 17, 18, 19]. Early works such as [15, 18] show that it is possible to use standard RL methods such as [12, 13, 20], but it can be time-consuming, and there are some types of tasks that these methods alone cannot solve. To combat this, other works have suggested the use of hindsight re-labeling [14, 21], which speeds up the process, but still requires a large amount of data to reach a successful trajectory [15].



(a) Sparse Reward (b) Brute Force Rewards (c) Ours

Fig. 2. **Visualization of Grid World Toy Example. (a)** shows the environment setup with a sparse reward. **(b)** shows the reward for each state once the entire environment has been solved using the bellman equation [22]. **(c)** represents WayEx where with a single demonstration, we compute a close approximation of the reward for each state along the path to the goal.

Imitation Learning. Learning from demonstrations, also known as imitation learning, is a common approach for solving goal conditioned tasks. Our approach uses ideas similar to prior works such as [5, 4, 2, 3, 1, 23]. These works take recorded successful episodes of a task and use them to aid in training the RL model. Several of these papers operate by adding successful demonstrations to the replay buffer [1, 23]; however, these papers require accurate knowledge of the action space. This type of information is only accessible in datasets specifically designed for robot training. In order to learn from other sources, such as videos, the field must evolve past this method. Additionally, aside from [5, 2], these works all require a large number of demonstrations. [5] can use a smaller number, but needs a task-specific reward function. [2] only uses one example, but requires different start states. [4] proposes pre-training the model and then fine tuning for each individual task, but their pre-training is data intensive and not always generalizable.

Inverse Reinforcement Learning. Another closely related area, Inverse Reinforcement Learning (IRL), uses a model to predict the reward values based on the state and action space. Methods such as [7, 9, 8], employ a hybrid approach involving both IRL and adversarial learning to solve a task with limited demonstrations. Similar to our method, IRL methods are useful because they do not require a defined reward function. Nonetheless, despite their claim of using a small number of demonstrations, these methods still require at least 50 demonstrations. Additionally, these methods struggle to generalize beyond the initial demonstrations. Our approach differs from IRL approaches because we do not require the training of a model to define our generalized reward, which leaves less room for error and requires less data.

Modified Sparse Reward. Several methods [24, 25, 26, 27] attempt to modify the sparse reward function in different ways to make learning more efficient. Despite employing a similar reward structure to ours and looking at the maximum of two different functions, [24] still requires a large number of demonstrations as well as access to the action space. Another way to learn with sparse rewards is to split multi-step tasks into several different tasks [25]. This allows the sparse rewards to be more frequent, but requires a precise definition of each auxiliary task which makes creating a

generalized model more difficult.

III. Method

A. Preliminaries

We formulate our problem as a Markov Decision Process (MDP) consisting of an [n]-tuple $(S, A, R, \tau_D, \gamma)$. The elements of this tuple are the state space S, the action space A, the reward function $R\colon S \times A \to \mathbb{R}$, the demonstration trajectory $\tau_D\colon (s_0^*, \ldots, s_N^*)$ and the discount factor γ. We will refer to a random episode trajectory as τ, where $\tau = (s_0, a_0, \ldots, s_N, a_N)$. N is the total number of states and actions to reach every state. A policy is represented as $\pi_\theta\colon S \to A$, with parameters θ. We use a sparse reward paradigm for our method, where the action space is unknown. A sparse reward is defined as a reward function R that receives a reward of 0 when in the goal state, g. At all other times the reward is -1. If g represents the goal state, and s_n and a_n are the n^{th} state and action respectively, then the sparse reward function R can be represented as

$$R(s_n, a_n) = \begin{cases} 0 & \text{if } s_n = g, \\ -1 & \text{if } s_n \neq g. \end{cases} \tag{1}$$

B. Overview

In this section, we describe our method WayEx for learning goal-conditioned skills from a single demonstration with no information about the demonstration's action space. An illustrative overview of our method is provided in Figure 2, which shows a much simpler grid world demonstration. The leftmost grid in Figure 2 represents the environment, where an agent must traverse the boxes to find the goal, which gives a sparse reward of 0. The middle grid shows the reward for each box using a bellman equation [22], which requires access to rewards for all states. Finally, the right grid shows our approach which traverses a single path and then recursively determines the reward of each state along the path. Access to one successful demonstration allows WayEx to determine the pseudo ground truth rewards for each box along the path.

WayEx uses a sparse reward, along with bellman's equation to compute the value for each waypoint along the demonstration path. During exploration, a new state obtains a reward if its distance from a known waypoint is less than a threshold d_{thresh} (captured by `is_prox_wp()`). We use Nearest Neighbors to determine the waypoint the new state is compared to. After training and achieving some success, we improve our method's ability to generalize to unseen start and goal states, by expanding on possible start and goal states. We present an algorithmic overview of WayEx in Algorithm 1, followed by a comprehensive breakdown of each step for a clearer understanding.

C. Proximal Waypoint

Each state of the environment can be represented as $s_i = \{p_1, p_2, \ldots, p_K\}$, $\forall$ i $\in (1, \ldots, N)$, where p_k represents an environmental parameter such as object pose and gripper velocity and K represents the number of environmental parameters. Note that each parameter, p_k, is a relative value, with respect to the object's location, instead of an absolute value with respect to the world coordinate system. This ensures better generalizability of our method. We use our policy π_θ to determine an action that allows us to reach an unseen state s_e. Following other reinforcement learning algorithms [13, 12], we add random noise to the action space, in order to increase the amount of exploration done during training. Once we have reached s_e we will determine if it is within close proximity of a waypoint along our demonstration trajectory τ_D. To do this we first use the Nearest Neighbor function to find the waypoint with the smallest total L2 distance between the parameters in the waypoint and the parameters in s_e as

$$s_t^* = \mathbf{NN}(s_e, \tau_D) = \underset{\forall s_i^* \in \tau_D}{\arg\min} \left\| s_e - s_i^* \right\|_2, \tag{2}$$

where $s_t^* \in \tau_D$ is the closest state to (or the proximal waypoint for) s_e from the states within the expert trajectory τ_D. For an agent to receive a reward at state s_e, the distance between the parameters of s_e and s_t^* should be less than a threshold. For instance, if $d_{\text{thresh}} = \{d_1, d_2, \ldots, d_K\}$ is the corresponding threshold for parameters between $s_e = \{p_1, p_2, \ldots, p_K\}$ and $s_t^* = \{p_1^*, p_2^*, \ldots, p_K^*\}$, then we compute the Boolean `hasProxWP` as

$$\texttt{is\_prox\_wp}(s_e, s_t^*) = \begin{cases} \text{True}, & \text{if } \|p_i^* - p_i\| \leq d_i, \forall i \in K, \\ \text{False}, & \text{otherwise.} \end{cases}$$

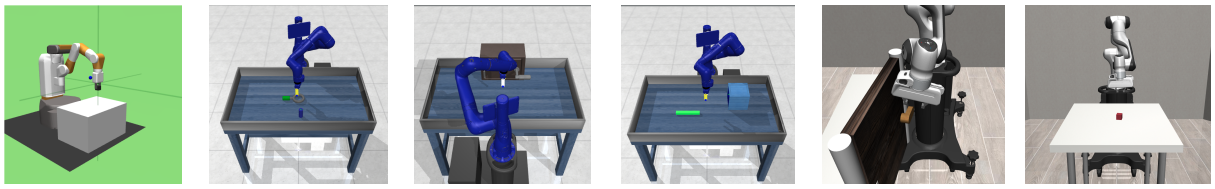
We define one d_{thresh} for each of the waypoints in τ_D. For instance s_t^* has its own threshold, d_t^*, and when s_t^* is the nearest neighbor to s_e, we use d_t^* as the threshold. In order to encourage progression, if `hasProxWP` is False 10 consecutive times for a waypoint s_t^*, then we increase d_t^* by ϵ $(= 0.001)$. We repeat this until we explore a point that falls within the threshold of the waypoint s_t^*.

Algorithm 1 WayEx, prior to expanding knowledge

```
τ_D = (s*_0, ..., s*_T) : A successful demonstration
π_θ : The policy that we will follow and update
while true do
    τ ← (s_0, a_0, ..., s_N, a_N) : an episode roll out
    for all s_n, a_n ∈ τ do
        s*_t ← NN(s_n, τ_D), where s*_t ∈ τ_D
        hasProxWP ← is_prox_wp(s_e, s*_t)
        r ← reward(hasProxWP, t, l_D, l_max)
        R ← max(r, γ * critic(s_{n+1}))
    end for
end while
```

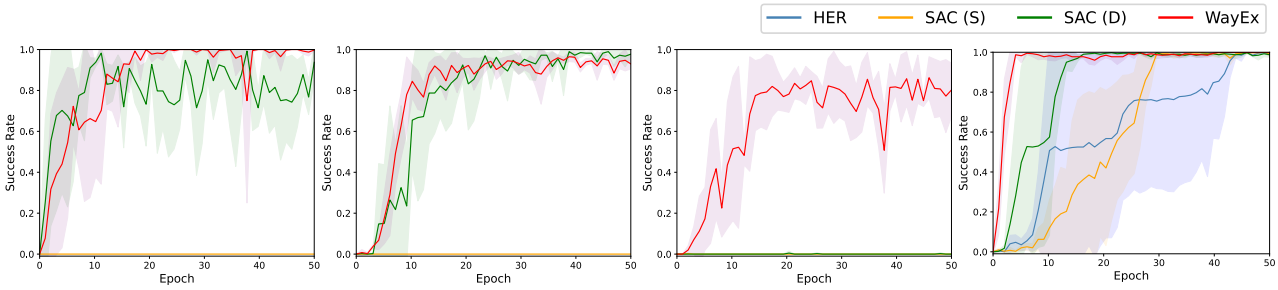
D. New Reward Function

Given the nearest neighbor waypoint, s_t^*, and the boolean result, `hasProxWP`, we can solve for the reward function, r, for our state action pair (s_e, a_e). l_{max} represents the maximum length of an episode, l_{D} represents the length of



(a) Pick and Place (b) Peg Assembly (c) Open Door (d) Peg Insertion (e) RS-Open Door (f) RS-Lift

Fig. 3. The environments that we experimented on with WayEx. We show results on 4 different tasks: (a) pick and place, (b) peg assembly, (c) open door and (d) peg insertion. These tasks are ideal because they have a clear definition of success and therefore a clear sparse reward. However, most of these tasks cannot be solved with sparse rewards alone.



(a) Door Open (b) Peg Insertion (c) Peg Assembly (d) Pick and Place

Fig. 4. **(a,b,c)** shows the results of the Meta World [28] environments when trained using SAC [13] and a batch size of 2048. **(a)** Open Door Task, **(b)** Peg Insertion Task, **(c)** Peg Assembly Task. **(d)** Pick and Place task is from OpenAI [15].

TABLE I

REQUIREMENTS FOR DIFFERENT BASELINES AND WAYEX IN TERMS OF STATE, ACTION AND NUMBER OF EXPERT DEMONSTRATIONS.

Pre-requisites	SAC (S)	SAC (D)	HER	SAC + RB	SAC + MCAC	AWAC	AWAC + MCAC	**WayEx**
Requires Action Space	✗	✗	✗	✓	✓	✓	✓	✗
Requires Pre-training	✗	✗	✗	✗	✗	✓	✓	✗
# Expert Demonstrations	0	0	0	1 or 100	1 or 100	1 or 100	1 or 100	**1**

the demonstration τ_D and t represents the time at which s_t^* occurs. We compute the proposed reward, r as

$$r = \begin{cases} \sum_{i=0}^{l_D - t} -\gamma^i & \texttt{hasProxWP}, \\ \sum_{i=1}^{l_{\max}} -\gamma^i & \text{not } \texttt{hasProxWP}. \end{cases} \tag{3}$$

If `hasProxWP` is false, we still want to account for the possibility that our state, s_e, is on a successful trajectory. To do this, we say that the final reward $R = \max(r, \gamma * \text{critic}(s_{e+1}))$. In order for this to work we need to warm up the critic. We do this by training it for 1000 time steps on just r before we include the critic reward. For more information on actor critic methods please refer to [13, 12].

E. Expanding Knowledge

Following the demonstration, WayEx teaches the policy how to solve the task from a fixed start and goal state. We now need to expand to every possible start and goal location. To achieve this, we slowly increase the number of possible start locations and goal locations, by adding random noise to the initial state space.

Given our current demonstration trajectory τ_D with states s_0^* and goal g^*, let $\mathcal{N}^*(\mu^*, \sigma^*)$ be the distribution representing the amount of noise we will add to the start state and goal state. We will set the mean, μ^*, to 0 and only increase the standard deviation σ^*.

At the start of training σ^* is set to 0. σ^* uses a modification strategy applied every 25 episodes. The updated value of σ^* is described as σ', where:

$$\sigma' = \begin{cases} \sigma^* + 0.001, & \text{if success rate} \geq 0.05, \\ \sigma^*, & \text{otherwise.} \end{cases} \tag{4}$$

IV. EXPERIMENTS AND RESULTS

A. Environment Setup

We use MuJoCo [31] to simulate our tasks. The implementation of our reinforcement learning algorithms was based on modified versions of the open-source code provided by stable-baselines3 [32] and our baselines were based on [24]. Although we used only one demonstration for each task, we varied the starting demonstration across different seeds to demonstrate the adaptability of our method to different initial demonstrations. To ensure robustness, we conducted each experiment four times with different seeds and present the mean and standard deviation of these seeds in our graphs. Each epoch consists of 40,000 simulated timesteps in MuJoCo. The pick and place environment, the robosuite

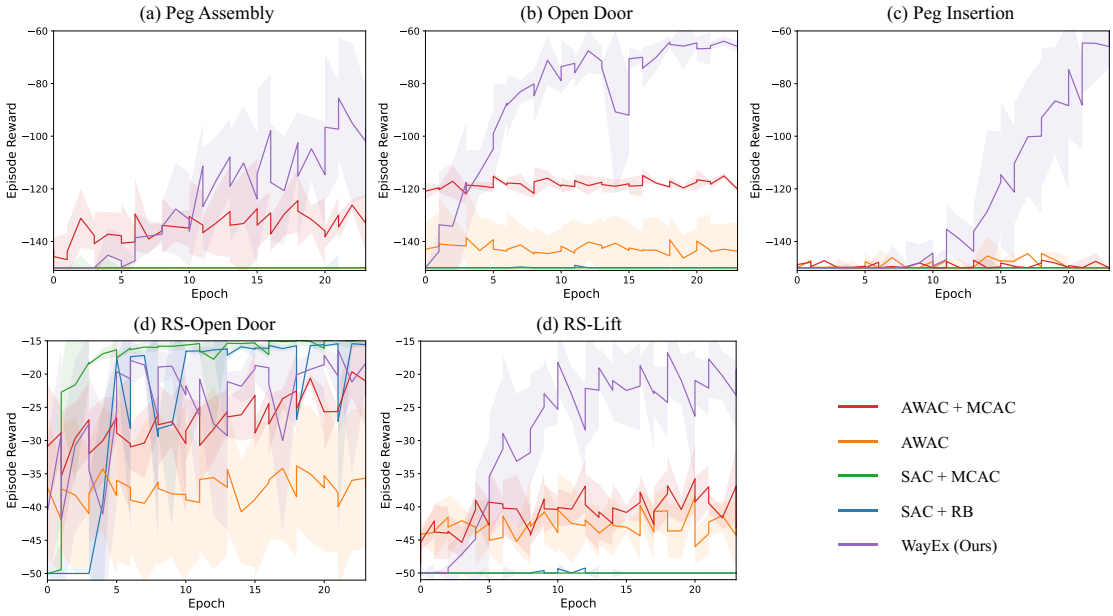


Fig. 5. This figure shows the results of several baselines when the are given one expert demonstration **(a,b,c)** shows the results of the Meta World [28] environments when trained using AWAC [29], MCAC [24], SAC + RB and AWAC + MCAC. **(d,e)** shows the results of these same baselines on the robosuite tasks [30]

door environment and the robosuite lift environment have 50 time steps per episode, and the remaining environments have 150 time steps per episode. We pre-train AWAC for 25000 timesteps.

B. Tasks

WayEx is trained on six distinct tasks, each designed to showcase the robot's capability to accomplish simple goal-conditioned objectives. The pick and place environment is from the OpenAI Fetch tasks [15], the next three (open door, peg insertion, and peg assembly) tasks are from Meta World [28]. The final two tasks, Robosuite (RS)-Open Door and RS-Lift come from [30]. Images of these tasks can be seen in Figure 3. Our tasks are: **(a) Pick and Place:** The task is to grasp a box and move it towards a goal in the air. **(b) Peg Assembly:** The task is to pick up a round nut and then place the round part over a peg. **(c) Open Door:** The task is to grasp a door handle and then open the door until it reaches a goal location. **(d) Peg Insertion:** The task is to pick up a peg and insert it into a hole. **(e) RS-Open Door:** The task is to grasp a door handle and then open the door very slightly. **(f) RS-Lift:** The task is to pick up a block and lift it into the air.

C. Baslines

We test our method (Table I) against following baselines:

- **SAC**. This baseline uses the Soft Actor Critic (SAC) algorithm [13] as the reinforcement learning algorithm. It can be initialized in three ways: (1) Sparse Reward **(SAC (S))**, (2) Dense Reward **(SAC (D))**, (3) SAC + Replay Buffer **(SAC + RB)**: we initialize the replay buffer with expert demonstrations.
- **Hindsight Experience Replay (HER)** is a well known technique [14], which uses the previous experiences to learn overtime.
- **SAC+MCAC**. This baseline uses the soft actor critic (SAC) algorithm [13] as well as Monte Carlo augmented Actor-Critic [24].
- **Advantage Weighted Actor Critic (AWAC)** follows the methods described in [29].
- **AWAC+MCAC** uses [29] along with a modified reward as describe in [24].

D. Results

We evaluate several different combinations of baselines each of which have different requirements in comparison to our method.

1) No Action Space: First we look at methods that do not require access to the action space of an expert demonstration in order to learn. In Figure 4(d), we analyze the performance of our method compared to other conventional reinforcement learning algorithms for the pick and place task. These graphs reveal two significant observations: (1) WayEx exhibits a remarkable acceleration in the learning process with just a single demonstration, over SAC. (2) WayEx demonstrates a considerably lower standard deviation compared to the other algorithms. We hypothesize that in general, methods relying on sparse rewards requires a degree of luck. For that, the environment must be explored extensively until a reward is obtained, allowing the method to learn the task. However, WayEx circumvents this by guiding the agent towards the goal, irrespective of the initial start point.

In Figure 4(a,b,c), we examine the outcomes of training three Meta World environments [28] using the SAC algorithm. These results are compared against a sparse reward and an episode-specific dense reward. Notably, we encountered difficulties in implementing hindsight experience replay with these environments. The Figure 4(a) represents the

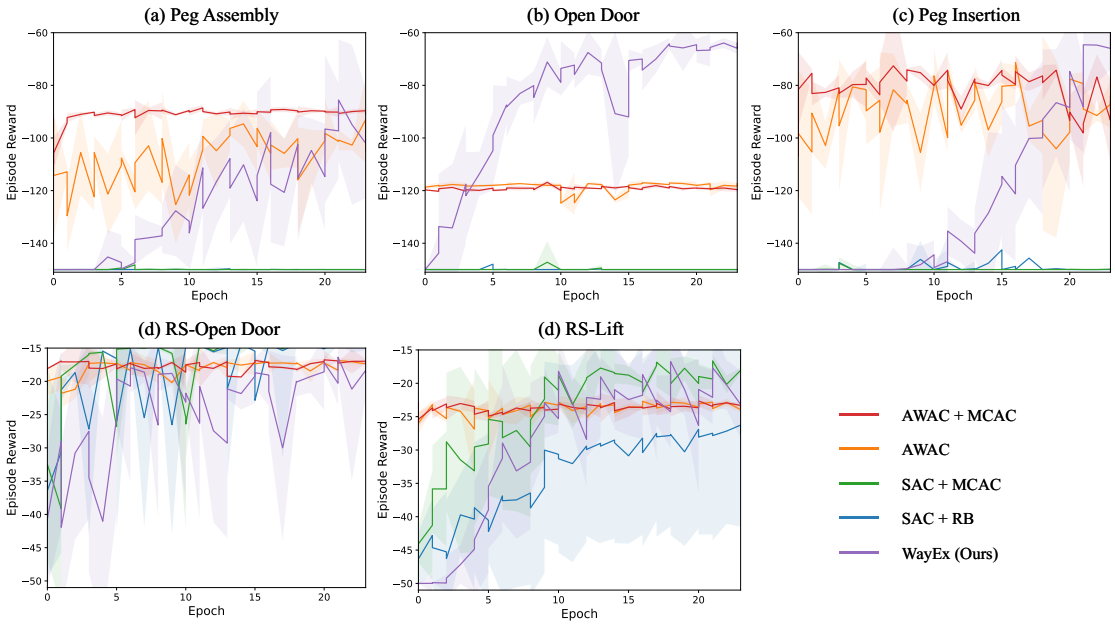


Fig. 6. This figure shows the results of several baselines when they are given one hundred expert demonstrations **(a,b,c)** shows the results of the Meta World [28] environments when trained using AWAC [29], MCAC [24], SAC + RB and AWAC + MCAC. **(d,e)** shows the results of these same baselines on the robosuite tasks [30]. Our method continues to use only one demonstration.

results of the open door task, (b) displays the results of the peg insertion task, and (c) showcases the peg assembly task. Across all these environments, we observed that SAC was unable to solve the tasks effectively when utilizing sparse rewards. For the open door and peg insertion task WayEx performs similar or better than the dense reward. The dense rewards have been finely tuned to each task and can require a lot of trial and error to finalize, while WayEx is a general reward that can be applied to any method. Regarding the peg assembly task, we discovered that SAC was unable to solve the task even with the dense reward during the training period. This is due to increased complexity of the task, which results in a noisier reward signal. This task is more difficult because there are a greater number of objectives to be accomplished. However, our method proved capable of swiftly solving the task despite these challenges.

2) One Expert Demonstration: Next, we look at our method compared to baselines that require just 1 example. We compare against AWAC, AWAC + MCAC, SAC + RB and SAC + MCAC when just one expert demonstration is used. The results of this are shown in Figure 5. We find that for the three meta world tasks our approach significantly outperforms the other approaches. AWAC and AWAC + MCAC work in the Peg Assembly and Door Open tasks, but they are not able to solve the problem as well as our approach is. For the RS-Door Open task, we find that our approach performs very similar to all other baselines. This is likely because the task is very easy and requires only a small amount of data. The RS-Lift task results look similar to the MetaWorld results where our method significantly outperforms the others. This task was more difficult than others due to the rotation of the block being different in each episode, but our method is able to handle it nonetheless.

3) 100 Expert Demonstrations: Finally, we look at our method compared to the same baselines when the baselines use 100 expert demonstrations. The results are shown in Figure 6. Note that AWAC, which is the method that performs the best in these scenarios has to be pre-trained in addition to the online training. We find that although our method takes more online training time with just one demonstration, versus 100 demonstrations, our method performs equal to or better than the baselines in all of the tasks.

V. Conclusion

We present WayEx, a new approach that enables training reinforcement learning models using a single demonstration. Unlike other imitation learning methods, which typically rely on multiple demonstrations and access to detailed action information, WayEx can operate with limited information and single demonstration. In order to achieve this, we introduce a novel universal reward function and leverage a knowledge expansion technique that extends beyond initial start and goal states. This makes it highly suitable for learning tasks with minimal information across different environments. We show that WayEx is faster than standard reinforcement learning models, in cases where the rewards are sparse or dense, and showcase its ability to succeed where other approaches fall short. Additionally, we show that WayEx performs similar to or better than a variety of imitation learning methods when these methods use either one or one hundred expert demonstrations. In future research, we aim to explore the use of expansion for non-linear states and investigate the utilization of image-based state spaces rather than joint locations.

Acknowledgements: This work was partially supported by DARPA SAIL-ON (W911NF2020009) program and NSF CAREER Award (#2238769) to AS.

References

[1] Ashvin Nair et al. "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *CoRR* abs/1709.10089 (2017). arXiv: 1709.10089. URL: http://arxiv.org/abs/1709.10089.

[2] Tim Salimans and Richard Chen. "Learning Montezuma's Revenge from a Single Demonstration". In: *CoRR* abs/1812.03381 (2018). arXiv: 1812.03381. URL: http://arxiv.org/abs/1812.03381.

[3] Yuke Zhu et al. "Reinforcement and Imitation Learning for Diverse Visuomotor Skills". In: *CoRR* abs/1802.09564 (2018). arXiv: 1802.09564. URL: http://arxiv.org/abs/1802.09564.

[4] Karl Pertsch et al. "Demonstration-Guided Reinforcement Learning with Learned Skills". In: *CoRR* abs/2107.10253 (2021). arXiv: 2107.10253. URL: https://arxiv.org/abs/2107.10253.

[5] Xue Bin Peng et al. "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". In: *ACM Trans. Graph.* 37 (2018), p. 143.

[6] Kei Ota et al. "Deep Reactive Planning in Dynamic Environments". In: *CoRR* abs/2011.00155 (2020). arXiv: 2011.00155. URL: https://arxiv.org/abs/2011.00155.

[7] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *CoRR* abs/1606.03476 (2016). arXiv: 1606.03476. URL: http://arxiv.org/abs/1606.03476.

[8] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. "SQIL: Imitation Learning via Regularized Behavioral Cloning". In: *CoRR* abs/1905.11108 (2019). arXiv: 1905.11108. URL: http://arxiv.org/abs/1905.11108.

[9] Justin Fu, Katie Luo, and Sergey Levine. *Learning Robust Rewards with Adversarial Inverse Reinforcement Learning*. 2018. arXiv: 1710.11248 [cs.LG].

[10] Christopher Watkins and Peter Dayan. "Technical Note: Q-Learning". In: *Machine Learning* 8 (May 1992), pp. 279–292. DOI: 10.1007/BF00992698.

[11] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[12] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning." In: *ICLR*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

[13] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

[14] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[15] Matthias Plappert et al. "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research". In: *CoRR* abs/1802.09464 (2018). arXiv: 1802.09464. URL: http://arxiv.org/abs/1802.09464.

[16] Soroush Nasiriany et al. "Planning with Goal-Conditioned Policies". In: *CoRR* abs/1911.08453 (2019). arXiv: 1911.08453. URL: http://arxiv.org/abs/1911.08453.

[17] Minghuan Liu, Menghui Zhu, and Weinan Zhang. *Goal-Conditioned Reinforcement Learning: Problems and Solutions*. 2022. arXiv: 2201.08299 [cs.AI].

[18] Richard Sutton et al. "Horde : A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction Categories and Subject Descriptors". In: vol. 2. Jan. 2011.

[19] Meng Fang et al. "Curriculum-guided Hindsight Experience Replay". In: *Neural Information Processing Systems*. 2019.

[20] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: (2013). cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. URL: http://arxiv.org/abs/1312.5602.

[21] Yevgen Chebotar et al. "Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills". In: *CoRR* abs/2104.07749 (2021). arXiv: 2104.07749. URL: https://arxiv.org/abs/2104.07749.

[22] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[23] Tom Le Paine et al. "Making Efficient Use of Demonstrations to Solve Hard Exploration Problems". In: *CoRR* abs/1909.01387 (2019). arXiv: 1909.01387. URL: http://arxiv.org/abs/1909.01387.

[24] Albert Wilcox et al. *Monte Carlo Augmented Actor-Critic for Sparse Reward Deep Reinforcement Learning from Suboptimal Demonstrations*. 2022. arXiv: 2210.07432 [cs.LG].

[25] Martin A. Riedmiller et al. "Learning by Playing - Solving Sparse Reward Tasks from Scratch". In: *CoRR* abs/1802.10567 (2018). arXiv: 1802.10567. URL: http://arxiv.org/abs/1802.10567.

[26] Alexander Trott et al. "Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards". In: *CoRR* abs/1911.01417 (2019). arXiv: 1911.01417. URL: http://arxiv.org/abs/1911.01417.

[27] Xingyu Lu, Stas Tiomkin, and P. Abbeel. "Predictive Coding for Boosting Deep Reinforcement Learning with Sparse Rewards". In: *ArXiv* abs/1912.13414 (2019).

[28] Tianhe Yu et al. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement

Learning". In: *Conference on Robot Learning (CoRL)*. 2019. arXiv: 1910.10897 [cs.LG].

[29] Ashvin Nair et al. *AWAC: Accelerating Online Reinforcement Learning with Offline Datasets*. 2021. arXiv: 2006.09359 [cs.LG].

[30] Yuke Zhu et al. "robosuite: A Modular Simulation Framework and Benchmark for Robot Learning". In: *arXiv preprint arXiv:2009.12293*. 2020.

[31] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control." In: *IROS*. IEEE, 2012, pp. 5026–5033. ISBN: 978-1-4673-1737-5. URL: http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

[32] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.