IMITATION LEARNING VIA DIFFERENTIABLE PHYSICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing imitation learning (IL) methods such as inverse reinforcement learning (IRL) usually have a double-loop training process, alternating between learning a reward function and a policy and tend to suffer long training time and high variance. In this work, we identify the benefits of differentiable physics simulators and propose a new IL method, i.e., Imitation Learning via Differentiable Physics (ILD), which gets rid of the double-loop design and achieves significant improvements in final performance, convergence speed, and stability. The proposed ILD incorporates the differentiable physics simulator as a *physics prior* into its computational graph for policy learning. It unrolls the dynamics by sampling actions from a parameterized policy, simply minimizing the distance between the expert trajectory and the agent trajectory, and back-propagating the gradient into the policy via temporal physics operators. With the physics prior, ILD policies can not only be transferable to unseen environment specifications but also yield higher final performance on a variety of tasks. In addition, ILD naturally forms a single-loop structure, which significantly improves the stability and training speed. To simplify the complex optimization landscape induced by temporal physics operations, ILD dynamically selects the learning objectives for each state during optimization. In our experiments, we show that ILD outperforms state-of-the-art methods in a variety of continuous control tasks with Brax, requiring only one expert demonstration. In addition, ILD can be applied to challenging deformable object manipulation tasks and can be generalized to unseen configurations.

1 INTRODUCTION

In a variety of applications ranging from games to real-world robotic tasks (Ho & Ermon, 2016; Fu et al., 2018; Zeng et al., 2020), imitation learning (IL) is popularly applied. However, collecting high-quality expert data is expensive, and existing IL methods tend to suffer long training time, unstable training process, high variance of learned IL policies, and suboptimal final performance.

Classical behavioral cloning (BC) methods learn policies directly from labeled data, but often suffer the covariate shift problem. This problem can be tackled in DAGGER (Ross et al., 2011) by interacting with the environment and querying experts online, which however requires significant human effort to label the actions. Other IL methods mainly include inverse reinforcement learning (IRL), adversarial imitation learning (AIL), and combinations of them. IRL learns a reward function to match expert demonstrations (Ziebart et al., 2008; Dadashi et al., 2021; Hoshino et al., 2022), and AIL learns a discriminator to identify whether the action comes from an expert demonstration (Ho & Ermon, 2016; Kostrikov et al., 2019). However, both IRL and AIL learn an additional intermediate signal, which introduces three main limitations: 1) the intermediate signal learning leads to a double-loop training process, which means long training time and complex implementation; 2) the learning signal is a noisy and frequently updated moving target, and as a result, the policy learning tends to have a high variance; 3) the intermediate signal, e.g., the reward function in IRL, inevitably loses the rich information embedded in the trajectories, e.g., environment dynamics.

In this work, we propose a new approach to IL, named Imitation Learning via Differentiable Physics (ILD), which recovers expert behavior by exploiting the Differentiable Physics Simulator (DPS) (Freeman et al., 2021; Hu et al., 2019). Different from standard environments, DPS implements low-level physics operations with a differentiable function and allows the gradients to flow through the dynamics. ILD takes advantage of DPSs by considering the environment dynamics as a *physics prior* and incorporating it into its computational graph during back-propagation of the policy, such

Table 1: Useful Properties among IL Methods

Property / Method Family	IRL	AIL	ILD (ours)
Layers of training loop	Double-loop	Double-loop	Single-loop
Source of the learning signal	Reward function	Discriminator	Differentiable dynamics
Transferability in changing dynamics	Depends	No	Yes

that the learned policy fully captures both the expert demonstration and the environment specifications. To achieve this, ILD simply minimizes the state-wise distance of a rollout trajectory generated by a parameterized policy to the expert demonstration, which also gives a single-loop design and avoids learning intermediate signals. Nevertheless, the gradients of physics operators are highly non-convex, which often introduces a complex optimization landscape, and consequently, a naive implementation is often stuck in local minimum (Freeman et al., 2021). To alleviate this issue, we introduce a simple yet effective Chamfer-α distance for trajectory matching. For each state in the rollout trajectory, instead of exactly matching the corresponding expert state, we dynamically select the easiest local goal as the optimization target and gradually proceed to the harder ones as training progresses. Chamfer-α distance naturally forms a curriculum learning setup, simplifies the optimization task, and eventually gives better final performance.

A short comparison of some useful properties of the IL methods can be found in Table 1. In contrast to the IRL and AIL methods, ILD does not introduce new intermediate signals and therefore requires no switching between policy learning and intermediate signal learning. In terms of the learning paradigm, IRL learns a reward function, AIL learns a discriminator, and ILD uses the differentiable dynamics which makes the learned policy aware of the environment dynamics and transferable to unseen environment configurations.

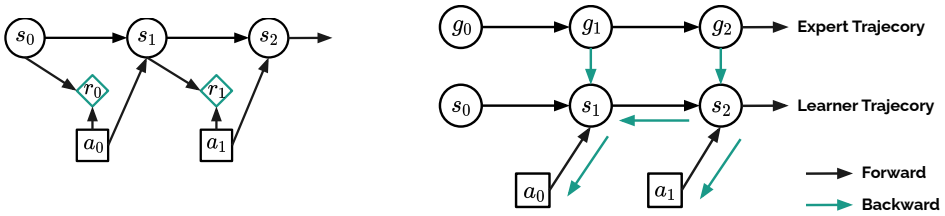
Empirically, we validate ILD on a set of MuJoCo-like continuous control tasks from Brax (Freeman et al., 2021) and a challenging cloth manipulation task. We show that ILD achieves significant improvements over the state-of-the-art IRL and AIL methods in terms of convergence time, training stability, and final performance. Given a fixed one-hour training time, ILD achieves 36% higher performance based on the normalized score over all the tasks and baselines.

2 Related Works

Imitation Learning. Classical imitation learning methods directly imitate the expert demonstrations via Behaviour Cloning (Pomerleau, 1991; Ross & Bagnell, 2010). However, they often suffer covariant shift problems due to insufficient expert training data. The modern approach GAIL (Ho & Ermon, 2016) uses the idea of generative adversarial networks to learn a discriminator that distinguishes between learner trajectories and expert trajectories. The agent explores the environment and learns to mimic the expert's trajectory. SAM (Sasaki et al., 2018) and DAC (Kostrikov et al., 2019) continue the idea of GAIL and address the sample efficiency problem. OPOLO (Zhu et al., 2020) also proposes a sample efficient learning from the observation (LfO) approach that allows for non-policy-based optimization.

Inverse Reinforcement Learning (IRL). IRL is a type of imitation learning that learns policies by recovering reward functions to match the trajectories demonstrated by experts (Argall et al., 2009). Early IRL methods such as MaxEntIRL (Boularias et al., 2011; Ziebart et al., 2008) minimize the KL divergence between the learner trajectory distribution and the expert trajectory distribution in the maximum entropy RL framework. However, those IRL methods often involve a double-loop learning process in which the outer loop learns the reward function and the inner loop solves the forward RL learning problem. AIRL (Fu et al., 2018) builds on the adversarial learning idea of GAIL by learning a reward function for reinforcement learning. Moreover, OPIRL (Hoshino et al., 2022) learns an off-policy reward function and solves the sample inefficiency problem. Recently, PWIL (Dadashi et al., 2021) proposes to learn the reward function by measuring the Wasserstein distance between the learner and the expert, achieving state-of-the-art results.

Differentiable Dynamics for Policy Learning. The differentiability of dynamics models has been explored to improve the stability and sample efficiency of policy learning. A commonly used paradigm is to learn a parameterized generative dynamics model by reconstructing the trajectory observations



(a) MDP Computation Graph

(b) ILD Computation Graph

Figure 1: Computational graph of MDP and ILD. A typical Markov decision process (MDP) includes a reward function to evaluate the performance of a policy and provide learning signals to the learner agent. In our approach, we eschew the reward function and use differentiable dynamics to dynamically match expert states and make gradients flow back into the action. This design provides two main benefits: 1) we move away from the double-loop design in IRL and AIL, avoiding the process of alternating between learning rewards and learning policies; 2) the analytic gradient from the dynamics provides richer information than a single reward number, which guides the improvement of actions in local regions, bringing less variance in training and better performance.

and train a policy by "imagined" trajectories with the learned generative model (Hafner et al., 2020; Ma et al., 2020; Hafner et al., 2021; Clavera et al., 2020). However, built upon a learned model, they suffer temporal accumulative model error and long training time by switching over two loops for model learning and policy learning (Yu et al., 2020). Recent advances in differentiable physics have shown their potential for policy learning by back-propagating the gradients through the physics operators (Hu et al., 2019; Lin et al., 2022; Huang et al., 2021; Freeman et al., 2021). Different from a learned model, differentiable physics provides a ground-truth understanding of the environment dynamics and naturally guarantees a good generalization. Nevertheless, the back-propagation through long temporal non-convex physics operators introduces a complex optimization landscape for policy learning, and as a result, a learned policy tends to be stuck at local minimums (Freeman et al., 2021).

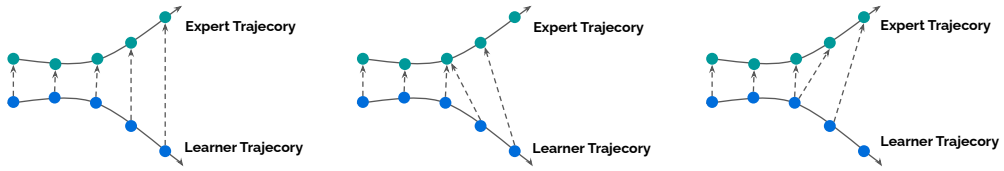
In contrast to existing methods, ILD avoids learning intermediate signals by computing the analytical learning gradient directly from the expert demonstration through differentiable physics. The analytical gradients carry rich information about both the expert intentions, i.e., the reward, and the specifications of the environment dynamics. Meanwhile, ILD dynamically selects local optimization goals for each state in the rollout trajectory, which gives a simpler optimization landscape for policy learning.

3 Imitation Learning via Differentiable Physics

We propose Imitation Learning via Differentiable Physics (ILD), which learns from expert demonstrations via differentiable physics without any additional intermediate signals, e.g., reward functions in IRL. We assume that the underlying transition function of the task is built on a set of physics rules so that the cumulative compound error is small compared to a dynamics model learned from data. Take a point-mass system as an example:

$$x_{t+1} = x_t + \Delta t \cdot v_t \qquad v_{t+1} = v_t + \Delta t \cdot \frac{f}{m}$$

where the force f is the action input to the system, v_t is the speed and x_t is the position of the point with mass m. The new position x_{t+1} can be computed analytically based on the physics properties such as the mass m. More importantly, such a system is differentiable and can carry the gradient from the output state x_{t+1} directly to the input force f. The same idea generalizes to more complex physics systems such as dynamics of deformable objects like cloth and liquid. For simplicity, we abuse the notation of physics and dynamics in this work. The gist of ILD is to consider the differentiable dynamics as a *physics prior* and incorporate it into the computational graph for policy learning. With differentiable dynamics, future states in the trajectory can exert influence on early actions. In this way, ILD can leverage the rich information from future states and learn a policy that is aware of the environment specifications. However, having a rich set of information does not necessarily lead to an optimal policy due to the complex optimization landscape through BPTT (Freeman et al., 2021). Therefore, we further decompose the optimization problem into many small and simple sub-steps by selecting suitable local learning such that each local goal can be effectively learned via differentiable dynamics.



(a) L2 Loss (b) Deviation Loss (c) Coverage Loss

Figure 2: Illustration of different loss functions. Both green dots and blue dots are the states in their trajectories. The L2 loss has a one-to-one matching, but the learning goal can be extremely difficult when later states deviate too much from the expert trajectory. The deviation loss only matches the closest state in the expert trajectory to reduce the difficulty of the learning goals, constraining the exploration space to be close to the expert trajectory. The coverage loss pulls the nearest states in the learner trajectory to be close to every state in the expert trajectory.

3.1 Differentiable Physics as Computational Graphs

The computation graph of our method can be found in figure 1. In a detailed view, our method ILD first rolls out the dynamics with the learner agent π_θ to interact with the environment to collect state trajectories. At each step t the learner policy $\pi_\theta(a_t|s_t)$ outputs the means and variances for all the action dimensions to sample actions, and the reparameterization trick (Kingma & Welling, 2014) is used to allow the gradient to flow through the sampling process. By iteratively unrolling the dynamics with the learner policy π_θ, we observe a trajectory τ_θ in the form of a list of states $s_{0:H}$ and a list of actions $a_{0:H}$. Treating the environment dynamics as a function $T(s_{t+1}|s_t, a_t)$, we can view the trajectory unrolling process as a temporal computation graph, $s_{1:H} = G(s_0, a_{0:H})$, where s_0 and $a_{0:H}$ are the inputs and $s_{1:H}$ denotes the outputs. Therefore, we can compute gradients regarding the action inputs. Since actions a_t are conditioned on the policy parameters θ, the entire computation graph G can be reduced to $s_{1:H} = G(s_0, \theta)$ that outputs a list of states $s_{1:H}$ and a list of actions $a_{0:H}$. Most importantly, by the virtue of differentiable dynamics, the entire computation graph G crossing multiple steps is fully differentiable.

In contrast to BC that minimizes the action distance $D_a(a \sim \pi_\theta || a^* \sim \pi_{\text{exp}})$ between the learner and the expert, we are minimizing the state distance $D_s(s \sim \tau_\theta || g \sim \tau_{\text{exp}})$ between the learner agent trajectory and the expert trajectory by differentiating through the temporal computation graph G. Such a design choice enables ILD with self-supervision in the unseen environment and hence addresses the covariant shift issue. To minimize the distance between the learner agent trajectories and the expert demonstrations, the first option is to apply a direct L2 loss between them, and back-propagate the gradient through the differentiable dynamics:

$$\arg\min_\theta \sum_{s\sim\tau_\theta} \sum_{t=0}^{T} (g_t - s_t)^2.$$

Based on the L2 loss objective function, each state g_t in expert demonstrations can be considered as a local learning goal at the time step t for the leaner agent to achieve. However, the objective function above suffers an issue of enforcing exact match between the state s_t in leaner policy trajectories and the state g_t. The corresponding learning goal g_t for each s_t may be impractical to achieve as the s_t and g_t can be far away at the beginning of the training state. However, such impractical goals exceed the capability that the differentiable dynamics can offer and hence often result in local optima.

3.2 Imitation Learning via Differentiable Physics

Considering the impractical learning goals in the L2 objective, we develop a new approach called Imitation Learning via Differentiable Physics (ILD). ILD considers the states in the imitation learning task as a set of unordered points and matches them to the expert demonstration. We introduce Chamfer-α loss for trajectory matching with the expert. Instead of selecting those faraway correct but impractical goals, ILD dynamically selects the nearest local goals to the demonstrated states, which gives a simpler optimization landscape. Specifically, Chamfer-α loss can be separated into two parts, deviation loss and coverage loss.

Algorithm 1 Imitation Learning via Differentiable Physics

Require: I Optimization Iteration
Ensure: The best estimated policy
1: Collect $J = 1$ expert demonstrations.
2: Initialize the stochastic policy as π_θ.
3: Pretraining π_θ using Behaviour Cloning.
4: **for** optimization iteration $i = 1 \cdots I$ **do**
    # Evaluate and optimize policy
5: Roll out trajectories τ_θ
6: Compute loss function L:

$$L = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s \in \tau_\theta} \|g_t - s\|_2^2 + \alpha \frac{1}{|\tau_\theta|} \sum_{s_t \in \tau_\theta} \min_{g \in \tau_{\text{exp}}} \|g - s_t\|_2^2$$

7: Update the policy π_θ with analytical gradient $\bigtriangledown_\theta L$.
8: **end for**
9: **Return** the policy π_θ.

Deviation loss. The Equation (1) shows the deviation loss function L_{d} for selecting the suitable goals. For each state $s \in \tau_\theta$, we treat it as a local optimization problem and set the learning goal to the nearest state $g \in \tau_{\text{exp}}$. The intention is to select the easier goals for the agent to follow. Concretely, this helps to constrain the gradient scale such that we can obtain a more stable optimization process during the BPTT with physics operators. The summation term measures the deviation loss between the learner policy roll-outs and expert demonstration. The intuition is that the learned policy should produce states similar to those from the expert policy and not deviate too much. We present the deviation loss as follows:

$$L_{\text{d}} = \frac{1}{|\tau_\theta|} \sum_{s_t \in \tau_\theta} \min_{g \in \tau_{\text{exp}}} \|g - s_t\|_2^2. \tag{1}$$

However, deviation loss alone may cause the "state collapse" issue. For example, it is possible that all $s_t \in \tau_\theta$ are close to a small subset of τ_{exp} with low deviation cost. As a result, deviation loss provides no coverage to the expert trajectory and is hence sub-optimal. Therefore, the ultimate goal is to learn π_θ that covers all the states of a trajectory τ_{exp} and at the same time stays close to the expert states.

Coverage loss. To ensure all states in the expert trajectory are covered by the learner policy, we introduce an extra coverage loss in Eqn. 2:

$$L_{\text{g}} = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s \in \tau_\pi} \|g_t - s\|_2^2. \tag{2}$$

Intuitively, the coverage loss guarantees that each state in the expert trajectory be close to the rollout trajectory generated by the learner policy. This naturally alleviates the "state collapse" issue introduced by the deviation loss. Combining the deviation loss and coverage loss, we introduce Chamfer-α loss function for ILD:

$$L_{\text{Chf-}\alpha} = L_d + \alpha L_g = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s \in \tau_\theta} \|g_t - s\|_2^2 + \alpha \frac{1}{|\tau_\theta|} \sum_{s_t \in \tau_\theta} \min_{g \in \tau_{\text{exp}}} \|g - s_t\|_2^2. \tag{3}$$

Although derived from different objectives, Eqn. 3 resembles the Chamfer distance for measuring the distance between two sets, which is widely used in computer vision (Butt & Maragos, 1998; Ma et al., 2010; Chen et al., 2021). Thus, we name our loss as Chamfer-α, which balances the ratio between deviation and coverage loss with a hyper-parameter α. In our experiments, we observe that having a lower α can shape the learned policy to imitate the expert at a global level and hence has a faster convergence. However, there is a trade-off between the convergence speed and final performance as the larger deviation factor converges slower but the final performance can be better. A visual illustration of Chamfer-α loss is shown in figure 2.

We summarize our method in Algorithm 1. We first collect $J = 1$ episode expert demonstration and use the standard supervised behavior cloning to bootstrap the policy learning using the expert demonstration. Next, we roll out the dynamics with a large batch of learner policy trajectories in

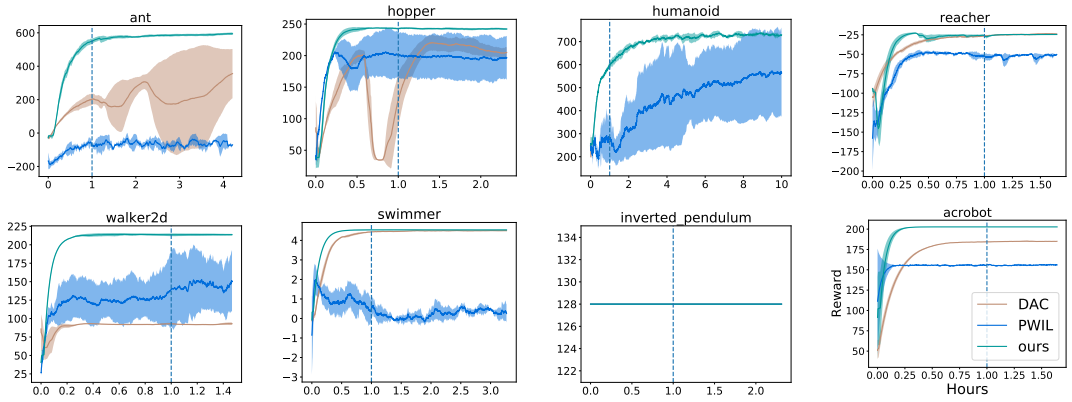


Figure 3: Relative wall time performance. We evaluate ILD on the Brax MuJoCo (Freeman et al., 2021) tasks and only one expert demonstration is provided to all the IL methods. The results show that our method generally outperforms the SOTA methods, with a much faster convergence speed, a more stable learning process, and smaller variance. The differentiable dynamics gives a single-loop training process and avoids the noisy intermediate signal. In addition, in difficult tasks such as ant, walker2d and humanoid, ILD outperforms the other baselines by a large margin given limited training time, because the differentiable physics helps to reserve more information embedded in the trajectories. Given an hour of training time (indicated by the vertical dashed line), we achieve 36% higher performance on average over all the tasks and baselines.

parallel to speed up the training. With the objective function (3) defined above, we compute the loss and update the policy parameters θ.

4 Experiment

We evaluate our method on Brax (Freeman et al., 2021) environments, providing a variety of differentiable MuJoCo-like continuous control tasks. In addition, we have developed a new robotic deformable object manipulation task, which requires hanging a piece of cloth on a stand, to demonstrate the generality of our approach to changing environment dynamics.

We compare our approach with two state-of-the-art IL methods, PWIL (Dadashi et al., 2021) from inverse reinforcement learning (IRL) and DAC (Kostrikov et al., 2019) from adversarial imitation learning (AIL). We follow their official published implementation and change the evaluation tasks to differentiable environments. For a fair comparison, we change their subsampling rate to use a single expert episode for training.

In our experiments, we aim to answer the following questions: 1) Can our method ILD recover expert behavior? 2) How fast does ILD converge? 3) Is ILD generalizable to complex deformable object manipulation tasks with changing dynamics? 4) What are the core parameters that influence performance?

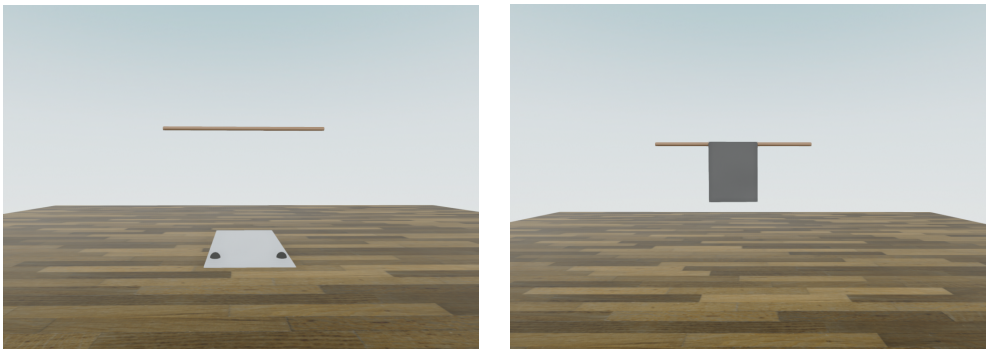
4.1 Brax Continuous Control Tasks

We evaluate our approach on the 8 Brax continuous control MuJoCo-like tasks: Ant, Hopper, Humanoid, Reacher, Walker2d, Swimmer, Inverted Pendulum, and Acrobot. For each task, we train a PPO (Schulman et al., 2017) agent to act as an expert. The episode length for all tasks is 128, and we collect only one expert episode for each task, which is used to train the IL agent. Each number reported in the result table is evaluated with 3 different seeds.

Results and Discussions. The overall performance is presented in Table 2 and the detailed training curves are given in figure 3. We observe that ILD outperforms the SOTA methods on 6 out of 8 tasks, and achieves comparable performance on the remaining 2 tasks. Also, with only one expert demonstration, ILD generally recovers the expert behavior and mostly achieves comparable performance. Specifically, we notice that the performance gain of ILD increases as the complexity of the task increases. For example, ILD achieves comparable performance with the baselines on

Table 2: Brax MuJoCo Task Results

	Ant	Hopper	Humanoid	Reacher	Walker2d	Swimmer	Inverted pendulum	Acrobot
DAC	393.57	220.54	256.63	**-21.52**	93.19	4.52	**128.00**	185.26
PWIL	-1.98	205.72	722.41	-47.19	205.78	1.98	**128.00**	157.03
ILD(ours)	**594.88**	**243.93**	**736.87**	-22.86	**214.17**	**4.54**	**128.00**	**202.74**
Expert	624.34	292.83	933.24	-22.49	289.14	4.29	128.00	200.80



(a) Initial State (b) Goal State

Figure 4: Deformable cloth manipulation. The task is to control two grippers (dark color) to hang the piece of cloth on the pole. Image (a) shows the initial state and image (b) shows the target state. This task is challenging because the dynamics are complex and the observation space is huge, with 1,736 dimensions.

the relatively easy task of Inverted Pendulum but outperforms the baselines on Ant environment. In addition, figure 3 shows that ILD has much lower variance and a more stable training curve compared with the SOTA methods. We believe both of the above advantages come from the benefits of differentiable physics. By back-propagating the gradients from states directly to the policy, ILD avoids the dynamic target introduced by the intermediate signals and stabilizes the training process; by considering the physics prior for policy learning, ILD obtains a policy that generalizes better to complex dynamics.

Moreover, although AIL and IRL methods benefit from the sample efficiency of the small number of interactions with the environment, they often suffer long training time due to the double-loop structure. Such a double-loop structure involves additional computations and slows down the training process. In contrast, ILD has a single-loop design and allows the policy to be optimized directly with the differentiable dynamics. In figure 3, we observe that ILD converges significantly faster than both DAC and PWIL with a much smoother and stable training curve.

4.2 Robot Cloth Manipulation

In this section, we test whether our approach can be generalized to a more complex robotic deformable object manipulation task, where a piece of cloth is to be hung on a pole. A visualization of this task can be found in figure 4. The main challenge of this deformable object manipulation task is the changing dynamics, where the testing dynamics are different from the demonstrating dynamics. Specifically, we add additional bias and noise to the input actions of the test environment, making the dynamics of the test environment different from the dynamics demonstrated by experts. As a result, we obtain an environment with noisy dynamics and fixed goals. In addition, the high-dimensional state space of deformable objects makes it hard to extract useful features for policy learning. To collect demonstrations, we use a handcrafted policy as the expert policy. Similarly, only one trajectory is used as the demonstration. Implementation details can be found in Appendix.

Results and Discussions. The results of the cloth manipulation can be found in figure 5. The results show that our method can learn a stable policy in changing dynamics under complex observation conditions. We outperform the other two baselines, completing the task with a success rate of 1. Specifically, ILD achieves fairly stable performance with low variance even if the environment dynamics are constantly changing. The success of ILD comes from the physics prior implicitly encoded in the learner policy, which learns a distribution of the environment specifications and can

Table 3: Brax MuJoCo Ablation Results

	Loss		Trunc Length				Deviation Factor			
	Chamfer-α	L2	1	10	30	100	0	0.2	1	5
ant	**583.77**	514.07	110.07	**583.77**	-15.50	-16.29	560.95	583.77	**594.88**	552.25
hopper	**242.27**	173.39	53.45	**242.27**	217.87	144.95	239.96	242.27	243.93	**248.76**
humanoid	**715.14**	542.94	331.27	715.14	**788.84**	355.09	704.96	715.14	**736.87**	710.12
reacher	-23.18	**-22.02**	-31.72	-23.18	-23.24	**-21.99**	-75.36	-23.18	**-22.86**	-22.95

recover the expert behavior even with unseen configurations. The other two baselines fail in the task, even though we have tried hard to tune the parameters such as reducing their learning rate. It is probably because they have learned a dynamics-coupled reward function or discriminator. When the dynamics change, the reward function/discriminator they have learned cannot be transferred, thus causing the failure. In contrast, our approach is not coupled with specific dynamics and thus has better generalization.

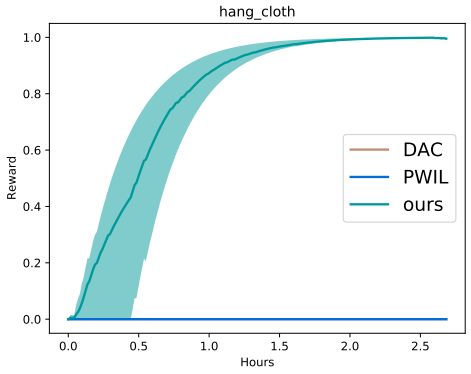


Figure 5: Deformable cloth manipulation Results. The task of manipulating the cloth has a very sparse reward: if the cloth is on the pole at the last step, the reward is 1; otherwise, all steps are 0. Therefore, the average reward can be interpreted as the task success rate. Compared to the expert demonstration environment, additional noise is added to the test environment, so the dynamics of the test environment change. Despite the dynamic change, our method ILD converges quickly and adapts to the new environment, while the other two baselines cannot be transferred in the dynamic change.

With the recent advances and growing interests in differentiable robot simulators, such as PlasticineLab (Huang et al., 2021), DiSECt (Heiden et al., 2021), and Scalable Diff (Qiao et al., 2020), we believe our method could be a good starting point for exploiting the differentiability of physics simulators to boost the development of robot learning algorithms.

4.3 Ablation Study

In this section, we discuss the core hyperparameters and algorithm choices that may influence performance. The results can be found in Table 3. The reported values in Table 2 are trained using the Chamfer-α loss with a truncation length of 10, batch size of 360, and deviation factor of 0.2.

Loss Function. We first compare the loss functions. We replace the original Chamfer-α loss with a simple step-wise L2 loss to compute the distance between the expert trajectory and the rollout trajectory. The results show that the performance of L2 loss is significantly lower compared to our method. This is because L2 loss simply performs a step-wise matching and ignores the actual distance between the states. As a result, this often introduces goals that are faraway from the current states and increases the optimization difficulty, especially through the BPTT over complex physics operators. Thus, it leads to suboptimal behaviors, which is a consistent observation with Freeman et al. (Freeman et al., 2021). In contrast, the proposed Chamfer-α distance selects the local goal that matches each state, thus reducing the complexity of learning and giving a smoother optimization landscape.

Truncation Length. We test with different gradient truncation lengths. As shown in the ablation results table, we find that short truncation lengths such as 1 usually lead to poor performance because in this case, the policy is trained by one-step signal and ignores the multi-step dynamics of the environment. However, large truncation lengths tend to lead to gradient explosion and in the end hinder the stability of the learning process. Nevertheless, when developing the cloth simulation environment, we observe that if with step-wise gradient normalization, the gradient explosion issue can be significantly alleviated and the performance of ILD can be generally improved. Empirically, a truncation length of 10 is a safe choice and often brings good performance.

Deviation Factor. We evaluate the influence of different deviation factors, which balance the deviation loss and coverage loss. We observe that the best performance is achieved with a deviation factor 1. However, we also observe that the smaller the factor is, the faster the global convergence

Table 4: Assumptions of Different Methods

	Training Phase				Testing Phase			
	RL	IRL	Planning	ILD (ours)	RL	IRL	Planning	ILD (ours)
Dynamics Model	✓	✓	–	✓	✗	✗	✓	✗
Expert Demo	✗	✓	–	✓	✗	✗	✗	✗
Reward Function	✓	✗	–	✗	✗	✗	✓	✗
Ground Truth State	✗	✗	–	✓	✗	✗	✓	✗
Observation	✓	✓	–	✓	✓	✓	✗	✓

will be, which introduces the trade-off between the final performance and the convergence speed. In our reported values, we use a deviation factor of 1, but the deviation factor 0.2 produces a comparable performance with a faster convergence speed, which could be considered for time sensitive applications. More details are available in the appendix.

4.4 Additional Discussions

We classify and compare different assumptions of different methods, including RL, IRL, planning, and ILD in Table 4. In contrast to RL, ILD and IRL do not require reward engineering, which is challenging and critical in many robotics and control tasks. In contrast to planning methods, ILD and RL methods are end-to-end policies that can handle image inputs during the testing phase, while typical planning methods have great difficulties in planning over image observations.

We then discuss the limitations of our method in the following aspects. First, there is a reduced the applicability of our method due to the assumption of differentiable dynamics. Many games, such as Atari games and board games, are not differentiable and thus cannot be applied. However, many valuable robotic tasks are indeed differentiable, because the underlying physics laws are differentiable. Trends in differentiable physics engines have emerged, such as PlasticineLab (Huang et al., 2021) for robotic deformable object manipulation, DiSECt (Heiden et al., 2021) for robotic knife cutting, and Scalable Diff (Qiao et al., 2020). If we can apply our approach to the above fields, we have taken a step towards the ultimate intelligent robot, despite the fact that more differentiable simulations will emerge in the future.

Finally, the domain mismatch between the simulation and the real world poses a real challenge. This challenge is precisely related to the sim-to-real problem, i.e., policies learned in simulations that are transferred to the real world. Since the sim-to-real problem is not our focus, we can apply existing methods to facilitate sim-to-real transfer, including: 1) Domain randomization (Akkaya et al., 2019; Tobin et al., 2018; Chebotar et al., 2019). For example, OpenAI (Akkaya et al., 2019) learns a policy in simulation only, but solves a Rubik's cube with real robot hands. The key idea is domain randomization, which randomizes textures, masses, object sizes, etc.. 2) Domain adaptation. It (James et al., 2019; Carlson et al., 2019) converts real-world observations into a similar form for policy trained in simulations. 3) System identification. Fast model identification (Zhu et al., 2017) and Tunenet (Allevato et al., 2020) identify environmental parameters and build more realistic dynamics models. In addition, differentiable dynamics has great potential to better estimate the simulator parameters using analytical gradients. We leave the sim-to-real problem using differentiable dynamics for future study.

5 Conclusion

In this work, we identify the benefits of differentiable dynamics and propose to use differentiable dynamics to learn IL agents. The core advantage of our approach is to move away from the traditional double-loop learning design and avoid the noisy intermediate learning signal. The differentiable dynamics provides us with a new type of IL algorithm and brings better performance and remarkably lower variance of the learned IL agent. The use of Chamfer-α distance enables dynamic selection of local targets, significantly reducing the learning difficulty and giving better performance. In future work, we will further address the sample efficiency issue by using small batches and short rollout trajectories. At the same time, we should target more challenging tasks, such as manipulation tasks of various robotic deformable objects. We conclude that our IL learning method has only a single learning loop, but outperforms other IL baselines. In addition, we demonstrate that our approach has great potential for more challenging but valuable robotic deformable manipulation tasks.

ETHICS STATEMENT

ILD is a general imitation learning method that requires differentiable dynamics. Working together with the sim2real methods, it may be effective for real robots. Caution should be taken when the method is deployed on real robots to prevent misuse of the proposed method. In summary, we have read the Code of Ethics and ensured that our work conforms to it.

REPRODUCIBILITY STATEMENT

We have tested the stability and reliability of our approach with multiple seeds on a variety of different tasks. Our code is included in the supplemental files. We will release the code upon publication.

REFERENCES

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Adam Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning*, pp. 445–455. PMLR, 2020.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, 2011.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

M Akmal Butt and Petros Maragos. Optimum design of Chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

Alexandra Carlson, Katherine A Skinner, Ram Vasudevan, and Matthew Johnson-Roberson. Sensor transfer: Learning optimal sensor effect image augmentation for sim-to-real domain adaptation. *IEEE Robotics and Automation Letters*, 4(3):2431–2438, 2019.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Siwei Chen, Xiao Ma, Yunfan Lu, and David Hsu. Ab Initio Particle-based Object Manipulation. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.071.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *ICLR*, 2020.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal Wasserstein imitation learning. *ICLR*, 2021.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. *NeurIPS Datasets and Benchmakrs*, 2021.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *ICLR*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ICLR*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ICLR*, 2021.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL `http://github.com/google/flax`.

Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *RSS*, 2021.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *NeurIPS*, 2016.

Hana Hoshino, Kei Ota, Asako Kanezaki, and Rio Yokota. OPIRL: Sample efficient off-policy inverse reinforcement learning via distribution matching. *ICRA*, 2022.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *ICLR*, 2021.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *ICLR*, 2019.

Xingyu Lin, Zhiao Huang, Yunzhu Li, Joshua B Tenenbaum, David Held, and Chuang Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *ICLR*, 2022.

Tianyang Ma, Xingwei Yang, and Longin Jan Latecki. Boosting Chamfer matching by learning chamfer distance normalization. In *ECCV*, 2010.

Xiao Ma, Siwei Chen, David Hsu, and Wee Sun Lee. Contrastive variational model-based reinforcement learning for complex observations. In *Proceedings of the 4th Conference on Robot Learning*, 2020.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. Scalable differentiable physics for learning and control. In *ICML*, 2020.

Arthur George Richards. *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology, 2005.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample efficient imitation learning for continuous control. In *ICLR*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3482–3489. IEEE, 2018.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *NeurIPS*, 2020.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020.

Shaojun Zhu, Andrew Kimmel, Kostas E Bekris, and Abdeslam Boularias. Fast model identification via physics engines for data-efficient policy search. *arXiv preprint arXiv:1710.08893*, 2017.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. *NeurIPS*, 2020.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

APPENDIX

A EXPERIMENTS

A.1 DEVIATION FACTOR

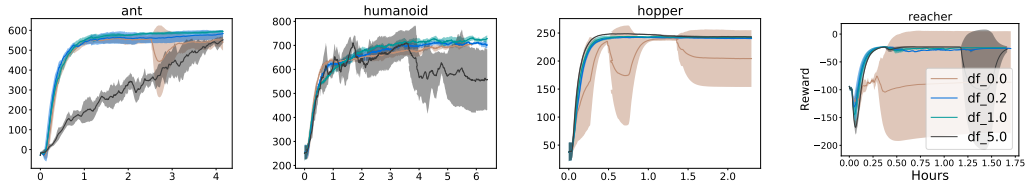


Figure 6: The training curves with different deviation factors. Setting the deviation factor to 0 or 5 will make the training unstable. However, ILD is less sensitive to moderate deviation values such as 0.2 to 1.

In figure 6, we show more details about the deviation factor. In general, if we do not force the learner state to be close to the expert state, i.e., set the deviation factor to 0, then the training process is unstable and tends to be suboptimal. On the other hand, if we focus too much on local state matching and set the deviation factor to 5, the learner policy tends to be conservative and unstable. Between these two, ILD is robust to deviation factors from 0.2 to 1 and does not vary much. In addition to this, if we compare 0.2 with 1.0, the smaller deviation factor learns slightly faster than the larger value, however, as a trade-off, its final performance is lower.

A.2 ABLATION STUDY

In Table 5, we show the results of ablation studies for eight Brax tasks, and a variant ILD-no-BC. The variant ILD-no-BC is our method that does not use expert actions for supervised learning to initialize the neural network. It shows that our proposed method does not rely on BC. without BC, ILD still achieves comparable performance in most tasks. The only problem is that the ILD without BC is

Table 5: Brax MuJoCo Ablation Results

		Loss		Trunc Length				Deviation Factor			
	ILD-no-BC	Chamfer-α	L2	1	10	30	100	0	0.2	1	5
ant	594.66	**583.77**	514.07	110.07	**583.77**	-15.50	-16.29	560.95	583.77	**594.88**	552.25
hopper	253.33	**242.27**	173.39	53.45	**242.27**	217.87	144.95	239.96	242.27	243.93	**248.76**
humanoid	657.66	**715.14**	542.94	331.27	715.14	**788.84**	355.09	704.96	715.14	**736.87**	710.12
reacher	-20.66	-23.18	**-22.02**	-31.72	-23.18	-23.24	**-21.99**	-75.36	-23.18	**-22.86**	-22.95
walker2d	213.26	209.00	**240.33**	72.13	**209.00**	203.53	61.77	214.00	209.00	**215.90**	214.43
swimmer	4.50	**4.57**	4.56	3.70	**4.57**	4.53	4.56	4.53	**4.57**	4.51	4.53
inverted_pendulum	128.00	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**	**128.00**
acrobot	70	**201.60**	194.67	201.67	201.60	**202.07**	202.06	201.67	201.6	202.06	**202.27**

less stable than the BC version on Acrobot. However, we believe that this does not affect the final conclusion. The ablation results of the other four tasks agree with the conclusion that the gradient truncation length cannot be too small or too long and Chamfer-alpha performs better than L2 loss in an overview.

A.3 Reinforcement Learning and Planning Baseline

Table 6: Brax MuJoCo RL and Planning Results

	Ant	Hopper	Humanoid	Reacher	Walker2d	Swimmer	Inverted pendulum	Acrobot
SHAC	-325.93	25.50	208.86	-17.23	88.73	4.60	128.00	160.23
CEM-MPC	-281.45	274.32	569.28	-139.19	276.08	10.60	128.00	154.71
ILD (ours)	594.88	243.93	736.87	-22.86	214.17	4.54	128.00	202.74
Expert	624.34	292.83	933.24	-22.49	289.14	4.29	128.00	200.80

We added another additional reinforcement learning baseline, SHAC (Xu et al., 2021) that exploits differentiable dynamics and planning baseline CEM-MPC (Richards, 2005). We implemented SHAC in JAX following the official implementation of PyTorch. Shown in Table 6, ILD achieved better results than SHAC on 6 out of 8 tasks for the same number of environmental interactions. We observed that SHAC is sensitive to hyperparameters and that hyperparameters have to be adjusted on a case-by-case basis. Given the time constraints, we used the same set of hyperparameters in all experiments. However, SHAC is a pure reinforcement learning algorithm, which requires an ground truth reward function, whereas the imitation learning method ILD does not. The baseline CEM-MPC accesses to the dynamics model and ground truth state during the test time, while ILD only uses observations. In the other tasks which only have image observations as input, planning-based methods have difficulties running. Therefore, we used the newly added results as a reference only and not as comparable baselines.

A.4 Multiple Expert Demonstrations

Table 7: Brax MuJoCo ILD with Multiple Demonstration Results

	Ant	Hopper	Humanoid	Reacher	Walker2d	Swimmer	Inverted pend	Acrobot
ILD-16	**641.36±6.09**	283.96±1.17	838.83± 26.87	**-20.91±0.68**	382.73±31.95	5.07±0.02	**128.00±0.00**	**202.73±0.05**
ILD-top1	633.80±9.21	**294.10±1.12**	**912.93±40.98**	-29.36±0.07	**419.83±14.15**	**5.11±0.01**	**128.00±0.00**	**202.73±0.05**
Expert	661.85±6.59	309.19±13.48	962.37±17.55	-5.99±1.45	421.45±8.00	5.18±0.09	128.00±0.00	202.64±0.19

To take advantage of more expert demonstration data, we add additional experiments comparing the ILD trained with 16 demonstrations to the ILD trained with the best of the 16 trajectories. Our results show that the ILD with only one trajectory actually outperforms the version with multiple trajectories. This is because ILD can successfully exploit expert information and therefore have high sample efficiency. In addition, the expert arguments contain uncertainties. Suboptimal trajectories actually hinder the overall performance. In conclusion, given its high sample efficiency, it is sufficient for ILD to use only one high-quality expert demonstration.

A.5 ILD IMPLEMENTATION DETAILS ON BRAX

In contrast to the IRL and AIL methods, our method ILD has only one policy network consisting of three MLP layers with Swish activation. The number of their hidden neurons is 512, 256, and the corresponding action dimension of the task, respectively. We clip the gradients with a maximum gradient norm value of 0.3 to regularize the learning process. To speed up the convergence, we use a batch size of 360 on an NVIDIA A100 graphics card. The deviation factor α and gradient truncation length are set to 1 and 10, respectively. We train our policy network with an Adam optimizer with a learning rate of 0.001 for 5,000 updates. The entire script is written by Flax (Heek et al., 2020) and JAX (Bradbury et al., 2018). For a fair comparison, all methods use the same amount of computational resource.

B CLOTH MANIPULATION

B.1 CLOTH SIMULATION DETAILS

Our cloth simulator is written in Jax and developed on top of the Taichi (Hu et al., 2019) implementation. As shown in figure 4, a piece of cloth is lying on the ground and the goal is to put this cloth on a pole by controlling two black grippers. The state of the cloth consists of 288 key points in the shape of $(288, 6)$, where the 6 dimensions are position and velocity. The underlying physics engine is built on Hooke's law, as shown below:

$$f_i = \sum_j -k(||x_i - x_j||_2 - l_{ij})(x_i - x_j)$$

$$v_{t+1} = v_t + \Delta t \cdot \frac{f}{m}$$

$$x_{t+1} = x_t + \Delta t \cdot v_{t+1}$$

where f_i is the force at the i_{th} point, j refers to the j_{th} neighbor, x_i is the position of the i_{th} point, and l_{ij} is the rest distance. In general, the longer the stretching distance, the higher the resistance force. By averaging the forces of all neighbors, we can calculate the next state of the point. In more detail, we set Δt to 2e-3 and repeat the above update equation 50 times for each robot action input. Thus, the dynamics of the deformable object is accumulated through time and the computational graph is long. To alleviate the gradient explosion and gradient vanishing problems, we normalize the gradients at each step of the backpropagation process.

B.2 ILD IMPLEMENTATION DETAILS

. We develop a cloth dynamics engine in JAX following the implementation of Taichi (Hu et al., 2019). The observation space for this task has 1,736 dimensions and consists of 288 key nodes on the cloth and 2 gripper states. The action space consists of 6 dimensions that control the speed of the two grasps. We assume that the two grippers have grabbed the two corners of the cloth. To facilitate the evaluation, we define a reward function that is 1 if the cloth is on the pole at the last step and 0 otherwise. This reward function also indicates the success rate of the training agent. The episode length for this task is 80 and a single expert demonstration is provided for all methods. We use the same implementation as the Brax environment with 3 MLP layers. In the complex task setting, we reduce the batch size from 360 to 50 due to hardware memory limitations. The learning rate is set to $1e^{-4}$ and the rest is the same.