Imitation Learning via Differentiable Physics

Siwei Chen, Xiao Ma, Zhongwen Xu
1Sea AI Lab 2National University of Singapore
siwei-15@comp.nus.edu.sg
{max,xuzw}@sea.com

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, 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 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, and AIL learns a discriminator to identify whether the action comes from an expert demonstration. 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

The code can be found here: https://github.com/sail-sg/ILD

Preprint. Under review.
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 |

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) [8, 16]. 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 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 naïve implementation is often stuck in local minimum [8]. 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 [8] 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 [23, 25]. However, they often suffer covariant shift problems due to insufficient expert training data. The modern approach GAIL [14] 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 [27] and DAC [19] continue the idea of GAIL and address the sample efficiency problem. OPOLO [31] 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 [1]. Early IRL methods such as MaxEntIRL [2] 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 [9] builds on the adversarial learning idea of GAIL by learning a reward function for reinforcement learning.
Moreover, OPIRL [15] learns an off-policy reward function and solves the sample inefficiency problem. Recently, PWIL [7] 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 and train a policy by “imagined” trajectories with the learned generative model [10, 22, 11, 6]. 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 [29]. Recent advances in differentiable physics have shown their potential for policy learning by back-propagating the gradients through the physics operators [16, 20, 17, 8]. 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 [8].

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:

\[
\begin{align*}
x_{t+1} &= x_t + \Delta t \cdot v_t \\
v_{t+1} &= v_t + \Delta t \cdot \frac{f}{m}
\end{align*}
\]

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
Treating the environment dynamics as a function, the computation graph of our method can be found in Fig. 1. In a detailed view, our method ILD considers the states in the imitation learning task as a set of unordered points and matches them to the expert demonstration. We introduce 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 trajectory by differentiating through the temporal computation graph, we 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 \(\theta\), 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 learner agent to achieve. However, the objective function above suffers an issue of enforcing exact match between the state \(s_t\) in learner 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
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 \( \pi_\theta \).
3: Pretraining \( \pi_\theta \) using Behaviour Cloning.
4: for optimization iteration \( i = 1 \cdots I \) do
    # Evaluate and optimize policy
5:     Roll out trajectories \( \tau_\theta \)
6:     Compute loss function \( L \):
7:         \[ L = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s_t \in \tau_{\text{exp}}} \| g_t - s_t \|^2 + \alpha \frac{1}{|\tau_{\text{exp}}|} \sum_{s_t \in \tau_{\text{exp}}} \min_{g \in \tau_{\text{exp}}} \| g - s_t \|^2 \]
8:     Update the policy \( \pi_\theta \) with analytical gradient \( \nabla_\theta L \).
9: end for
10: Return the policy \( \pi_\theta \).

Chamfer-\( \alpha \) 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-\( \alpha \) loss can be separated into two parts, deviation loss and coverage loss.

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:

\[
L_d = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s_t \in \tau_{\text{exp}}} \| g_t - s_t \|^2. \tag{1}
\]

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 \( \tau_{\text{exp}} \) 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 \( \pi_\theta \) that covers all the states of a trajectory \( \tau_{\text{exp}} \) 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)

\[
L_g = \frac{1}{|\tau_{\text{exp}}|} \sum_{g_t \in \tau_{\text{exp}}} \min_{s_t \in \tau_{\text{exp}}} \| g_t - s_t \|^2. \tag{2}
\]

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-\( \alpha \) loss function for ILD:

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

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 \[4\[2\[1\]]\[5\]. Thus, we name our loss as Chamfer-\( \alpha \), which balances the ratio between deviation and coverage loss with a hyper-parameter \( \alpha \). In our experiments, we observe that having a lower \( \alpha \) 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-\( \alpha \) loss is shown in Fig. [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 parallel to speed up the training. With the objective function (3) defined above, we compute the loss and update the policy parameters \( \theta \).

4 Experiment

We evaluate our method on Brax 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 [7] from inverse reinforcement learning (IRL) and DAC [19] 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 [28] 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.

Implementation Details. 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 0.5 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
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  |

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.

Results and Discussions. The overall performance is presented in Table 2 and the detailed training curves are given in Fig. 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 the relatively easy task of Inverted Pendulum but outperforms the baselines on Ant environment. In addition, Fig. 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 Fig. 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 Fig. 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.

![Hang Cloth Results](image)

**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.

**Implementation Details.** We develop a cloth dynamics engine in JAX following the implementation of Taichi [16]. 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 $1 \times 10^{-4}$ and the rest is the same. More details of the cloth simulation can be found in Appendix.

**Results and Discussions.** The results of the cloth manipulation can be found in Fig. 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 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.

With the recent advances and growing interests in differentiable robot simulators, such as PlasticineLab [17], DiSECt [13], and Scalable Diff [24], 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-$\alpha$ 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-$\alpha$ 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     | Chamfer-α | L2     | Trunc Length | Deviation Factor |
|------------|-----------|--------|--------------|------------------|
|            | Loss      |        | 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 | -21.18 | -22.86 | -22.95 |

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. [8]. 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 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 Limitations

Our approach relies on differentiable dynamics, but not all tasks are naturally differentiable. Many games, such as Atari games and board games, are non-differentiable and therefore our approach does not apply. However, in the real world there are many valuable tasks that are indeed differentiable, such as robotic deformable object manipulation. In addition, a lot of new differentiable physics engines have emerged, such as PlasticineLab [17] for soft deformable objects, DiSECt [13] for knife cutting, and Scalable Diff [24] for cloth and soft objects manipulation. Another limitation is the gradient truncation length, where too short truncation length does not perform well, and too large can result in gradient explosion. However, if the differentiable dynamics normalizes its gradient in every step-wise transition function call, the gradient exploding issue can be significantly alleviated.

### 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.
Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD-2021-08-015T).

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
[2] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, 2011.
[3] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
[4] M. A. Butt and P. Maragos. Optimum design of Chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.
[5] S. Chen, X. Ma, Y. Lu, and D. Hsu. Ab Initio Particle-based Object Manipulation. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.
[6] I. Clavera, V. Fu, and P. Abbeel. Model-augmented actor-critic: Backpropagating through paths. *ICLR*, 2020.
[7] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal Wasserstein imitation learning. *ICLR*, 2021.
[8] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. *NeurIPS Datasets and Benchmarks*, 2021.
[9] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *ICLR*, 2018.
[10] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *ICLR*, 2020.
[11] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *ICLR*, 2021.
[12] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2020.
[13] E. Heiden, M. Macklin, Y. Narang, D. Fox, A. Garg, and F. Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *RSS*, 2021.
[14] J. Ho and S. Ermon. Generative adversarial imitation learning. *NeurIPS*, 2016.
[15] H. Hoshino, K. Ota, A. Kanezaki, and R. Yokota. OPIRL: Sample efficient off-policy inverse reinforcement learning via distribution matching. *ICRA*, 2022.
[16] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.
[17] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *ICLR*, 2021.
[18] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.
[19] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *ICLR*, 2019.
[20] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *ICLR*, 2022.
[21] T. Ma, X. Yang, and L. J. Latecki. Boosting Chamfer matching by learning chamfer distance normalization. In *ECCV*, 2010.
[22] X. Ma, S. Chen, D. Hsu, and W. S. Lee. Contrastive variational model-based reinforcement learning for complex observations. In Proceedings of the 4th Conference on Robot Learning, 2020.

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

[24] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Scalable differentiable physics for learning and control. In ICML, 2020.

[25] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In AISTATS, 2010.

[26] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In AISTATS, 2011.

[27] F. Sasaki, T. Yohira, and A. Kawaguchi. Sample efficient imitation learning for continuous control. In ICLR, 2018.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[29] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. MOPO: Model-based offline policy optimization. NeurIPS, 2020.

[30] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. CoRL, 2020.

[31] Z. Zhu, K. Lin, B. Dai, and J. Zhou. Off-policy imitation learning from observations. NeurIPS, 2020.

[32] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In AAAI, 2008.
Appendix

6 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 Fig. 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.

7 Cloth Simulation

Our cloth simulator is written in Jax and developed on top of the Taichi [16] implementation. As shown in Fig. 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:

\[
f_i = \sum_j k||x_i - x_j||_2^2 - l_{ij})(x_i - x_j)\]

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

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

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 \(\Delta t\) 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.