


Replication / ML Reproducibility Challenge 2022

# Reproducibility of Plan Your Target and Learn Your Skills: Transferable State-Only Imitation Learning via Decoupled Policy Optimization

Anonymous<sup>1</sup>, <sup>1</sup> Anonymous Institution

## Reproducibility Summary

**Scope of Reproducibility** – The authors of the paper introduce Decoupled Policy Optimization (DePo), a learning architecture that decouples the control policy into two distinct modules: a high-level state planner and an inverse dynamics model. They claim that DePo learns accurate state predictions with the best imitation performances. They also claim that DePo allows pre-training and co-training with less sampling cost for agents over various action spaces compared to other state-of-the-art training models.

**Methodology** – We used the author’s code to reproduce their experiments on three different simulation setups: grid world experiment, Mujoco, and NGSIM dataset, to verify the three claims made by the original paper. Extensive qualitative and quantitative results are analyzed to check the claims. All the experiments were run on a NVIDIA Geforce RTX 3080 Ti GPU.

**Results** – All three claims were supported by our reproduction results. Due to time and computation resource constraints, we were not able to train as many epochs as the original paper did. However, our reproduction work yielded remarkably similar results to the original paper and the proposed DePO method indeed outperformed the baseline methods in most cases. The results of our experiments suggested the correctness of the original claims yielded by the original experiments in the authors’ paper.

**What was easy** – For the MuJoCo experiments, the expert data collected by the authors of the original paper is extensive and easy to access for every experimental setup, which makes it relatively easy to train and test the models and reproduce the work.

**What was difficult** – The document for the code was sparse and not well written, and it took significant time to configure their code base for our environment. We tried contacting the authors of the article, but received no response. In addition, training took significant time and computational resources.

**Communication with original authors** – We emailed the authors of the paper with questions but received no response.

## 1 Introduction

With Imitation Learning (IL), agents can be trained from demonstrations by mimicking the expert’s behaviors without constructing hand-crafted reward functions. While information regarding both the state and the action are needed for accurate learning, typically the latter is only achieved via expert demonstrations. In most real-world demonstrations, the action data is not available. While state-only imitation learning (SOIL) methods exist, current methods only attempt to match the expert state sequences implicitly by determining feasible actions for each pair of consecutive states. However this methodology ignores understanding of the high-level target planning strategy of the expert, and therefore significantly reduces the generalizability of the learned control policy.

The authors attempted to model a generalized target planner that allowed transference to various action spaces and dynamics. They developed Decoupled Policy Optimization (DePo), an architecture that decouples a state-to-action policy as two modules: a state planner that generates the consecutive target state, followed by an inverse dynamics model that delivers action to achieve the target space.

## 2 Scope of reproducibility

This report describes our efforts to reproduce the work from the original paper *Plan Your Target and Learn Your Skills: Transferable State-Only Imitation Learning via Decoupled Policy Optimization* [1], which introduces Decoupled Policy Optimization (DePo) to explicitly decouple the policy as a high-level state planner and an inverse dynamics model to leverage demonstrations and improve the flexibility of the policy. The authors’ main claims regarding the robustness of DePo were written explicitly in the original paper as follows:

- *Claim 1:* DePo has a generalized ability of state planning and is able to predict the legal state transition on out-of-demonstrations states.
- *Claim 2:* DePo learns accurate state predictions while keeping the best imitation performances, i.e., DePo enables the agent to reach exactly where it plans to, and the state planner is accurate enough that it can be used for long (more than one)-step planning.
- *Claim 3:* DePo allows pre-training and co-training with less sampling cost for agents over various action spaces and dynamics but same state space, i.e., with the decoupled policy structure, DePo is able to learn the state planner once (either by pre-training or co-training) and is only required to retrain the inverse dynamics model for agents share the same state space.

In Section 4 we reproduced multiple experiments and designed new ones to demonstrate all the three claims.

## 3 Methodology

The original paper provides an open-source implementation of their code on GitHub: <https://github.com/apexrl/DePO> and we used this code and repeated the experiments. Our code can be found here: [https://github.com/Theochiro/EECS\\_553\\_Team\\_11](https://github.com/Theochiro/EECS_553_Team_11). The detailed model is described in Section 3.1. The dataset and hyperparameters we used are described in 3.2 and 3.3 respectively. In 3.4 and 3.5, we discussed our experimental setup and computational resources.

### 3.1 Model descriptions

The framework the original paper proposed is composed of three parts: the state planner  $h$  used for predicting the next state; the inverse dynamics model  $I$  used for calculating the action needed to reach the next state from the current state; and a generative adversarial training process. This last part is a composite of the proposed decoupled policy serving as the generator, and a discriminator  $D$  to compute the intermediate reward. The overall structure is shown in Fig 1.

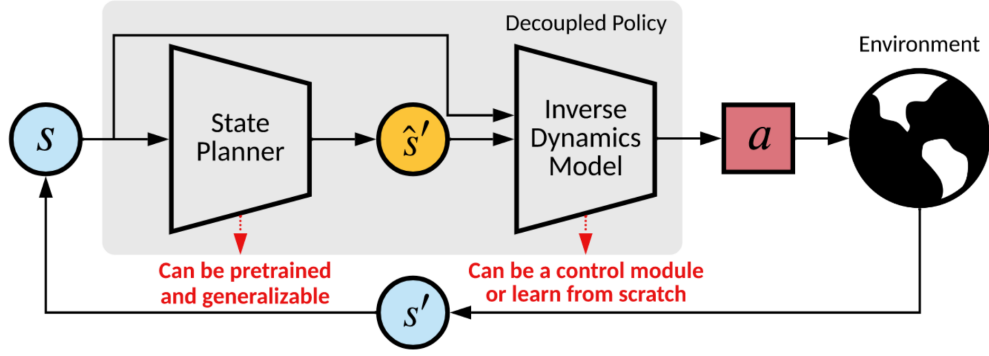


Figure 1. Overall Architecture of DePO [1]

The state planner module and the inverse dynamics module can be learned by supervised learning and a policy gradient method, named Decoupled Policy Gradient (DePG). Specifically, the inverse dynamics model can be learned online, pre-trained in advance, or simply provided as input in the form of a ground-truth function if known. The learning objective is to minimize the divergence between the inverse dynamics of a sampling policy  $\pi_B$  and  $I_\phi$  as shown below:

$$\min_{\phi} L^I = E_{(s,s') \sim \pi_B} [D_f(I_{\pi_B}(a|s,s') || I_{\phi}(a|s,s'))] \quad (1)$$

The state planner is constructed to predict the next target state given the current state information. A way proposed in the original paper to learn the parameterized module  $h_\psi$  was to minimize the divergence to match expert data:

$$\min_{\psi} L^h = E_{(s,s') \sim \Omega_B} [D_f(h_{\Omega_B}(s'|s) || h_{\psi}(s'|s))] \quad (2)$$

In addition to the supervised learning approach, the original paper also derived a policy gradient approach which treats the state planner module and the inverse dynamics module as an integral policy function. If we assume the reward is known and denote the state-action value function as  $Q(s, a)$ , then by chain rule, the policy gradient can be accomplished by:

$$\nabla_{\phi, \psi} L^{\pi} = E_{(s,a) \sim \pi} [Q(s, a)] \nabla_{\phi, \psi} \log \pi_{\phi, \psi}(a|s) \quad (3)$$

By assuming the inverse dynamics is deterministic given current state  $s$  and predicted next state  $s'$ , the policy gradient can be simplified as:

$$\nabla_{\psi} L^{\pi} = E_{(s,a,s') \sim \pi} [Q(s, a)] \nabla_{\psi} \log h_{\psi}(s'|s) \quad (4)$$

Therefore, the overall objective of the proposed algorithm can be summarized as:

$$\min_{\phi} L^I (\text{if } I \text{ needs to be learned}) \quad (5)$$

$$\min_{\psi} L^{\pi,h} = L^{\pi} + \lambda_h L^h \quad (6)$$

where  $\lambda_h$  is the hyperparameter for trading off the loss.

For implementation details, we used the same implementation as the original paper used. The value network and the discriminator are implemented as two-layer multilayer perceptrons (MLPs) with 256 hidden units and 128 hidden units respectively. The state predictor is a two-layer MLP with 256 hidden units and the inverse dynamics model is a four-layer MLP with 512 hidden units. The learning process is conducted using the Soft Actor-Critic (SAC) algorithm, and the optimizer used is Adam.

### 3.2 Datasets

The expert data we used in this work is the data collected by the original paper. The authors of the original paper trained a SAC agent to collect expert data of different dynamics models in the Mujoco simulator for model training and testing. They trained and tested DePo on the Inverted Pendulum, Hopper, Walker, Half Cheetah and Humanoid models in simulation. The data can be downloaded from their GitHub page.

### 3.3 Hyperparameters

The hyperparameters which yielded the best performance for DePo were provided in the original paper. To minimize extraneous variables in our reproducibility study we used the same hyperparameters in the training process of this work. The values of hyperparameters are listed in Table 1. Note the additional state planner coefficient  $\lambda_h$  is not listed since this value was changed by the authors for each model.

**Table 1.** Hyperparameters during experiments

Environments	Hopper	Humannoid
Trajectory maximum length	1000	
Optimizer	Adam	
Discount factor	0.99	
Replay buffer size	2e5	
Batch size	256	
State planner coefficient	0.1	0.01
learning rate	3e-4	
learning rate	3e-4	
learning rate	3e-4	
learning rate	1e-4	
learning interval (epochs)	10	
Gradient penalty weight	4.0	16.0
Reward scale	2.0	

### 3.4 Experimental setup and code

The instructions for setting up the code are provided in the GitHub page of the original paper. However, to reproduce the work, we had to do the following to make the code successfully run on our laptops: (a) Whereas the original paper discussed the robustness of DePo on multiple dynamics models, the code in their GitHub repository was only provided for one particular dynamics model. We made changes to the code so that it can be generalized to different .yaml files corresponding to the remaining models discussed

in the original paper. (b) The only working python version is python 3.7. There are dependency issues for all other versions. For detailed instructions, please refer to our GitHub page.

To verify the authors' claims we have conducted experiments measuring the following three metrics:

1. Average Return corresponds to learning efficiency
2. The Mean Square Error (MSE) between the state planner and the real state that the agent achieves in the environment
3. The Success Rate represents the percentage of movement without illegal transitions.

The authors verified their claims using the first two listed metrics. We are therefore using the same two metrics to reproduce their original results. The third metric was conceived to measure the robustness of DePo beyond the studies conducted by the original paper. Figures depicting the results of the above metrics are presented in Section 4.

### 3.5 Computational requirements

Experiments were reproduced on a laptop equipped with NVIDIA Geforce RTX 3080 Ti GPU. And it takes 100 epochs to train DePo, and 150 epochs to train the other two baseline methods BCO and GAIfO.

## 4 Results

To extensively verify the authors' claims, we conducted experiments in three simulation environments and datasets including:

1. Claim 1 (DePo is able to generalize to out-of-demonstration states) was verified via experiments on a grid world environment
2. Claim 2 (DePo has better imitation performance compared to the baseline methods of) was verified via Mujoco simulation comparing DePo to BCO [2] and GAIfO [3])
3. Claim 3 (DePo allows pre-training and co-training with less sampling cost for agents over various action spaces and dynamics but same state space) was verified via evaluation of training on the NGSIM I-80 dataset [4](a traffic surveillance recording dataset)

Our results validated all three claims of the authors' original paper. However, on complex dynamics models in Mujoco such as the Humanoid model, DePo yields poor results. Each experiment and its results will now be further discussed.

### 4.1 Results reproducing original paper

**Result 1 (Claim 1 verification via a grid world experiment):** — The claim of the generalized ability of DePo (claim 1) was evaluated via a grid world environment created by the original authors. In their experiment, expert demonstrations were first provided for an agent navigating from the bottom left corner (0,0) to the top right corner (5,5). Agents were then trained on three learning methods: Supervised DePo, Agnostic DePG, and DePo. These trained agents were then randomly placed on non-demonstration states (white locations) on the map and were tasked with navigating to the top right corner (5,5). Using their provided code, we reproduced their experiment for the above three training methods and compared our results with the depicted in the paper. The results are shown

below in Figures 2 and 3. Black arrows represent the actions taken in each state and yellow cells represent the cells visited by the expert while generating the demonstrations.

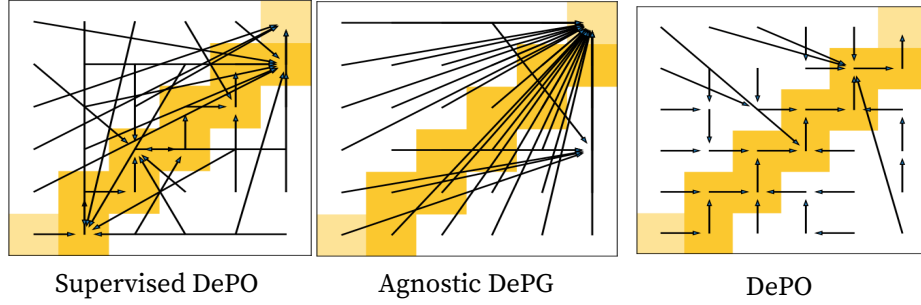


Figure 2. Reproduced Result for Grid World Experiment

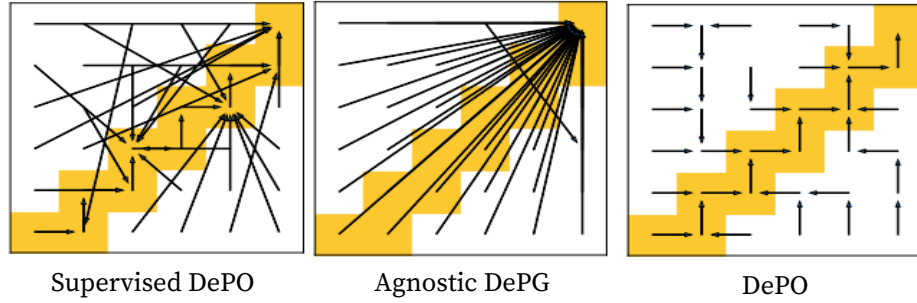
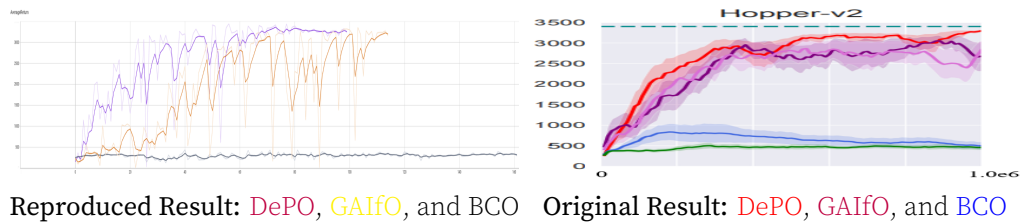


Figure 3. Original Results for Grid World Experiment

Although our reproduction work has some deference from the original work, the reproduction results indeed show that DePO performs well and is able to generate legal actions in most of the out-of-demonstration states, thus supporting claim 1.

**Result 2 (Claim 2 verification via a Mujoco experiment):** – In this result subsection, we reproduced the Mujoco experiment for DePo conducted by the original authors to verify that DePO has better imitation performance compared to baseline training methods. Using the authors’ original code, we conducted experiments on a Hopper model in simulation. We trained the Hopper on DePo, and two baseline methods: BCO, and GAIfo. The imitation performance was demonstrated by the learning curves, as shown in Figure 4. Due to the computation resource and time constraints, we only trained 100 epochs for DePO, 150 epochs for BCO, and 120 epochs for GAIfo. For a fair comparison, only the first 100 epochs were used for analysis. The reproduced and original results are shown in Figure 4.



Reproduced Result: DePO, GAIfo, and BCO    Original Result: DePO, GAIfo, and BCO

Figure 4. Average Return of Reproduced Results and Original Results

We can see the pattern matches the original plot: DePo achieves the highest return in

the same number of epochs, which shows DePO has better sample efficiency compared to BCO and GAIfO, leading to better imitation performance, which supports claim 2.

**Result 3 (Claim 3 verification via experiments on grid world experiment and NGSIM dataset)** – The original authors demonstrated the claim that DePO can have less sampling cost through pre-training and co-training (claim 3) using grid world experiment (described in Result 1) and NGSIM dataset respectively in the original paper. To reproduce their work, we trained the algorithms in these two settings as well, as shown in Figures 5 and 8.

First we verified the reduced sampling cost of DePO through pre-training using the grid world experiment. The agent was first trained with action space parameter  $k = 1$ . The parameter was then changed to  $k = 4$  for the transferring training stage. The metric used here is *Success rate* representing the percentage of movement without illegal transitions. The original plot and our reproduced plot with the same environment setting are shown in Figures 5 and 6.

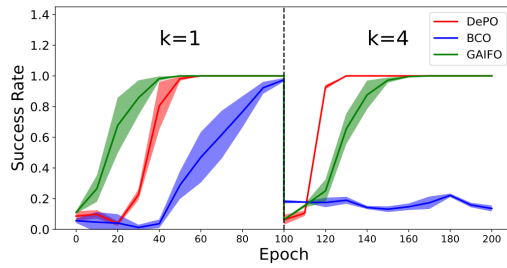


Figure 5. Reproduced Result

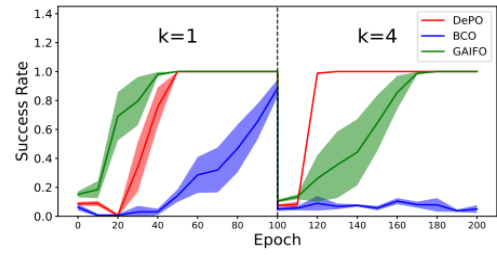


Figure 6. Original Result

As is shown, our reproduced result is nearly identical to that of the original plot, demonstrating faster convergence speed compared to the baseline methods, supporting the DePO’s advantage through pre-training.

Next, we verified the reduced sampling cost of DePO through co-training. The original paper demonstrated the co-training ability of DePO using NGSIM dataset. The authors used three different inverse dynamics functions named *Normal*, *Inverse*, *Transpose* to mimic different vehicles. Since all kinds of vehicles shared the same high-level state space, the authors of the original paper argued that DePO could utilize it by only learning the shared state planner through co-training, achieving much less sampling cost and the fastest convergence rate. We reproduced their experiments by training on the same dataset. Due to the time and computational resource constraints, we only trained 16 epochs for DePO and did not train the baseline GAIfO method. The original results and our reproduced result are shown in Figure 7 and Figure 8.

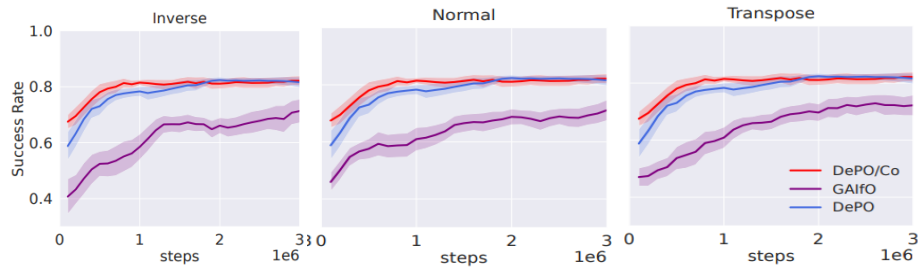
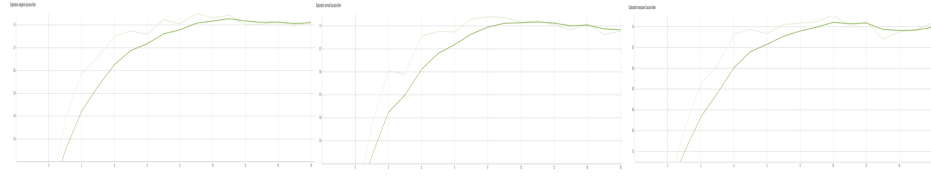


Figure 7. Original Results using the NGSIM Dataset. Plots are shown for Inverse, Normal and Tranpose.



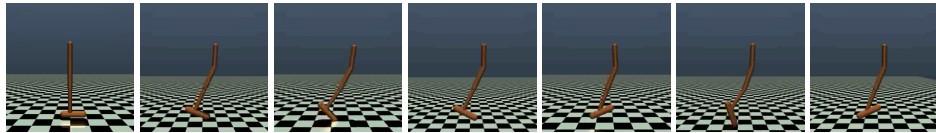
**Figure 8.** Reproduced Results using the NGSIM Dataset. Plots are shown for Inverse, Normal and Transpose. Due to technical limitations, the axis labels were not able to be recovered. However, they correspond directly with those found in fig 7.

As shown, our reproduced DePo Success Rate yields identical results to that of the original paper, thus verifying DePo’s reduced sampling cost in co-training. Since we have verified DePo’s reduced sampling cost in both pre-training and co-training, we validate claim 3 of the original paper.

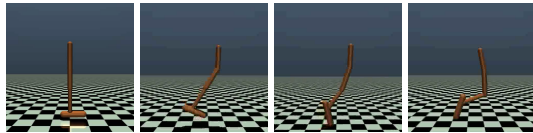
## 4.2 Results beyond original paper

In this section, we perform several experiments to compare the performance of DePo with other state-of-the-art methods for Imitation Learning such as BCO and GAIfo.

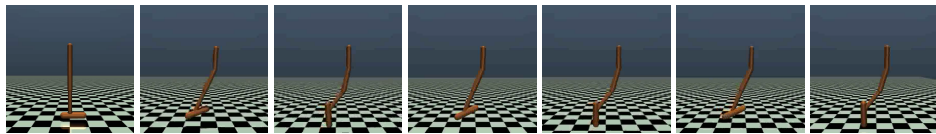
**Additional Result 1** – To qualitatively check the planning ability of the state planner, we illustrated the images of the imaged rollout states predicted by the learned state planner of DePo, compared with the rollout states predicted by the two baseline methods GAIfo and BCO, and the real reaching states achieved during interaction with the environment. In Figures 9, 10, 11, each image represents one second progressions for each imitation method. As is shown, with less training epochs (DePo: 100, BCO: 150, GAIfo: 120), DePo performs as well as GAIfo and better than BCO in terms of mimicking the real reaching states. (Note that BCO only shows 3 seconds because it prematurely collapsed in simulation.) This additional result further supports the claims that DePo has better sampling efficiency and more accurate state predictions compared to baseline methods.



**Figure 9.** Hopper Planner (DePo)



**Figure 10.** Hopper Planner (BCO)



**Figure 11.** Hopper Planner (GAIfo)



**Additional Result 2** – After completing the above section, we performed an additional experiment to test DePo robustness on complex models. Using the Mujoco environment, we measured the performance of DePo on the Humanoid platform. The results are shown in Figure 12. As is shown, the humanoid model collapses after only 3 seconds. Unlike stated in the paper, the that DePo does not seem to perform well for complex environments.

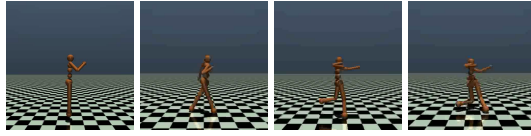


Figure 12. Humanoid Planner (DePo)

**Note:** Due to the page limit, the rest of the additional results are included in the Appendix.

## 5 Discussion

In this report, we reproduced several experiments within three different simulator environments: a Grid World environment, a MuJoCo environment, and an NGSIM environment. With these experiments, we were able to reproduce all the results of the original paper and included several additional ones. Although our reproduction work and the original work are not exactly identical, our results did support the authors' claims that DePO is able to generate reasonable actions in out-of-demonstration states, which verified claim 1. For claim 2, we conducted reproduction experiments in the Mujoco simulation platform and trained a Hopper robot using the code provided by the original authors. Our reproduction work shows DePO indeed has better learning efficiency and prediction accuracy compared to other two baseline methods, supporting claim 2. Lastly, we checked the pre-training and co-training ability of DePO through the reproduction experiments in both grid world experimentation and the NGSIM dataset. Our reproduced success rate plot was identical to the one in the original paper. Therefore, we also validated claim 3 of the original paper.

### 5.1 What was easy

For the MuJoCo experiments, the expert data collected by the authors of the original paper is extensive and easy to access for every experimental setup, which makes it relatively easy to train and test the models and reproduce the work.

### 5.2 What was difficult

The difficulties encountered in this reproducibility report were all related to the codebase. With the lack of documentation, as well as the tremendous amount of bugs encountered, most of the time was spent trying to debug the code and modify the original version into our improved codebase, which can be found [here](#).

### 5.3 Communication with original authors

We emailed the authors of the original paper with questions but received no response.

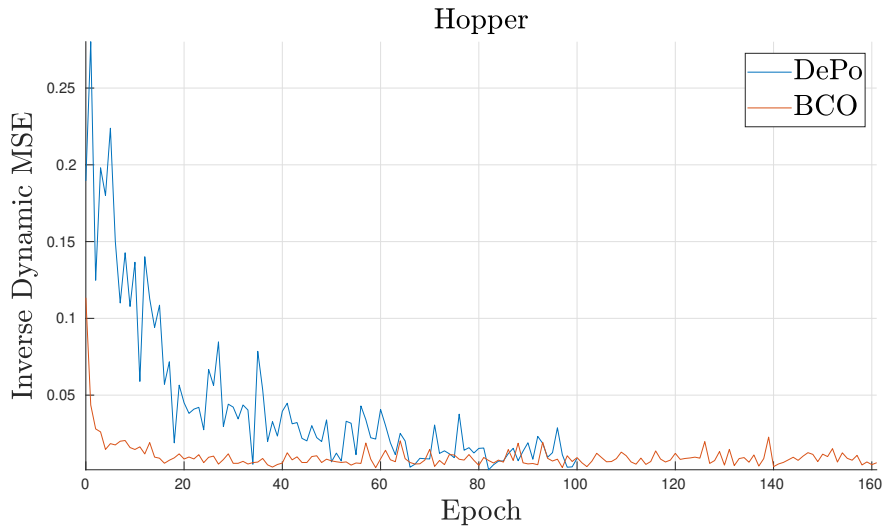
## References

1. M. Liu, Z. Zhu, Y. Zhuang, W. Zhang, J. Hao, Y. Yu, and J. Wang. "Plan Your Target and Learn Your Skills: Transferable State-Only Imitation Learning via Decoupled Policy Optimization." In: **Proceedings of the 39th International Conference on Machine Learning**. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 14173–14196.
2. F. Torabi, G. Warnell, and P. Stone. "Behavioral cloning from observation." In: **arXiv preprint arXiv:1805.01954** (2018).
3. F. Torabi, G. Warnell, and P. Stone. "Adversarial imitation learning from state-only demonstrations." In: **Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems**. 2019, pp. 2229–2231.
4. J. Halkias and J. Colyar. "Next generation simulation fact sheet." In: **US Department of Transportation: Federal Highway Administration** (2006).

## 6 Appendix

### 6.1 Additional Result 3

In Figure 13, included in the Appendix, we provide a plot of the inverse dynamics MSE for BCO and DePo during training for the Hopper experiment. We can clearly see that BCO outperforms DePo by achieving a very low MSE within 5 epochs compared to DePo, which needs 50 epochs to do so.



**Figure 13.** Plot of the Inverse Dynamics MSE for DePo and BCO

### 6.2 Additional Result 4

In this part, Figure 15 shows the performance of the different DePo methods when training with randomized initial states. Black arrows represent the actions taken in each state and yellow cells represent the cells visited by the expert while generating the demonstrations. We can clearly see that the performance is much better than with fixed initial state as shown in Figures 3 and 2.

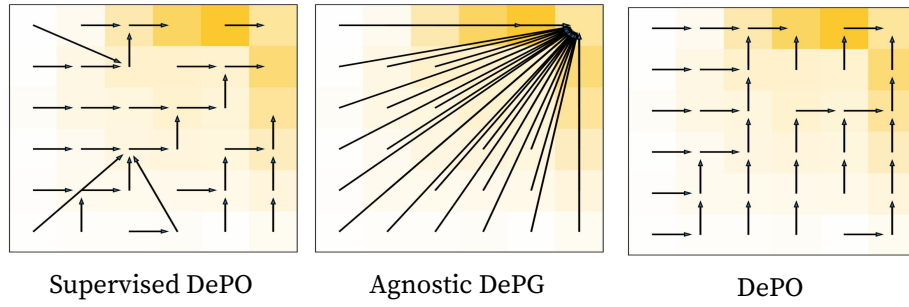


Figure 14. Policy Results for randomized training

### 6.3 Additional Result 5

Here, we plot the evaluation heat maps of each method as well as the expert demonstration for both randomized training and normal training. In randomized training, the agent is trained with a random initial state in the grid world as opposed to normal training where the agent always starts at the top left cell. the expert actions are chosen to follow a straight line to the bottom right cell which is the end goal state. Clearly, when trained with a deterministic initial state, DePo outperforms GAIfo and BCO while in randomized training, this is not the case.

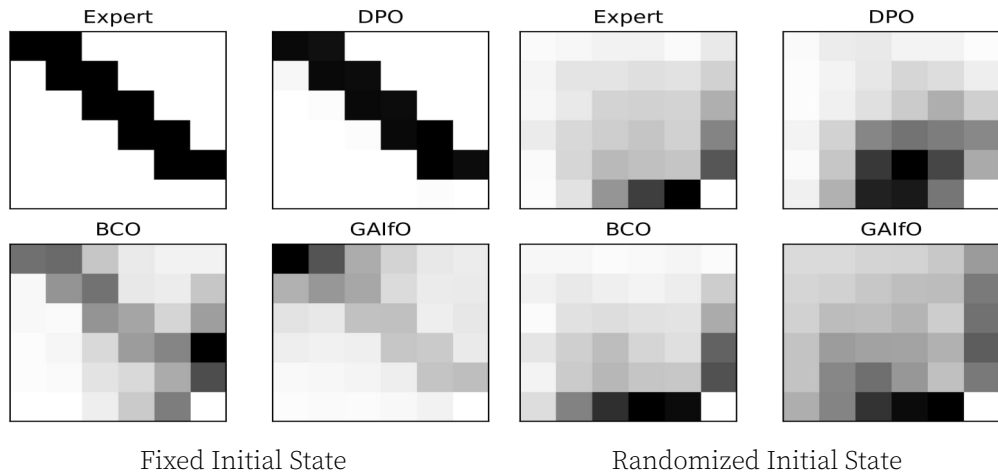


Figure 15. Heat Maps

### 6.4 Additional Result 6

In Figure 16, we compare the performance of DePo, GAIfo, and BCO using different metrics for a fixed and randomized initial state. We can see that GAIfo outperforms DePo in both cases using these different metrics.

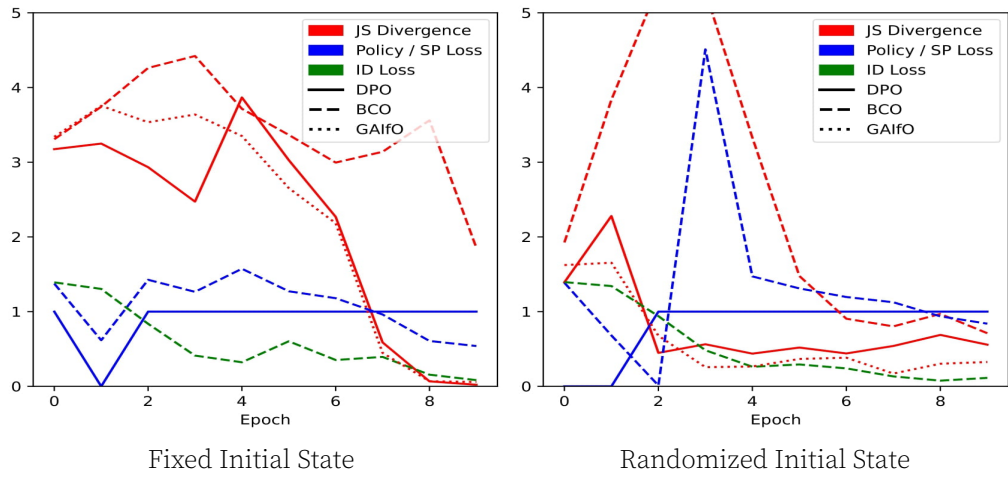


Figure 16. Comparison using several metrics

### 6.5 Additional Result 7

In Figure 17, the top two are the expert demonstrations. The bottom figures are the DePo reconstruction. The left two are supervised learning and the right half is supervised learning. As expected, training supervised DePo yields better performance over unsupervised DePo learning.

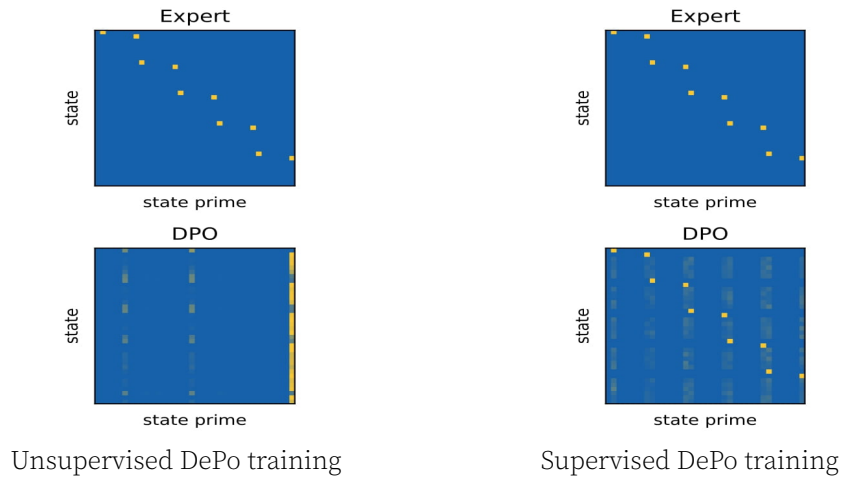


Figure 17. Comparison using several metrics