

APPENDIX

NULL COUNTERFACTUAL FACTOR INTERACTIONS FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

TABLE OF CONTENTS

A	Reproducibility Statement	16
B	Flow Diagram	16
C	Visualization of HInt using Ground Truth or NCH interactions	16
D	Simulated Nulling	16
E	Filter Criteria	17
F	Investigating ϵ_{null}	18
G	Network Architectures	19
G.1	Pointnet Architectures	19
G.2	Graph Architectures	19
G.3	Transformer Architectures	20
H	Environment Details	20
H.1	Spriteworld	20
H.2	Robosuite	20
H.3	Air Hockey	21
H.4	Franka Kitchen	21
I	Training Details	21
J	Hindsight Sampling Ablation	22
K	Vision Experiments	23
L	Additional Comparisons	24
M	Rollout Visualizations	25

A REPRODUCIBILITY STATEMENT

We provide the implementation of NCII, HInt, and other baselines in the appendix code. The installation instructions, detailed settings, and configurations for the data generation process for the benchmarks and datasets can be found in Appendix H and the appendix code. The training details, including hyperparameter settings, experimental setups, and platforms, are provided in Appendix I.

B FLOW DIAGRAM

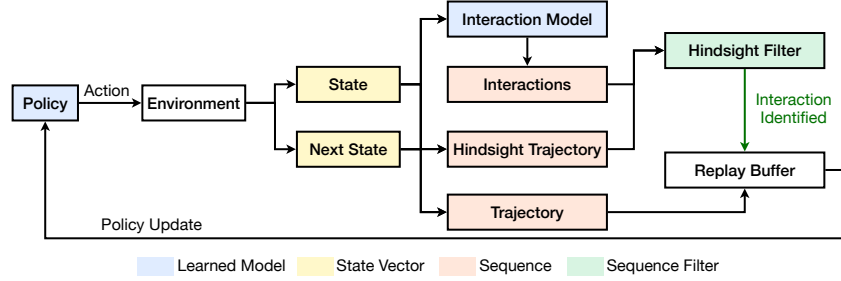


Figure 6: The data flow for the HInt method. An interaction matrix \mathbb{B} is generated at each time step, and at the end of each trajectory is used to filter non-interacting trajectories.

C VISUALIZATION OF HINT USING GROUND TRUTH OR NCII INTERACTIONS

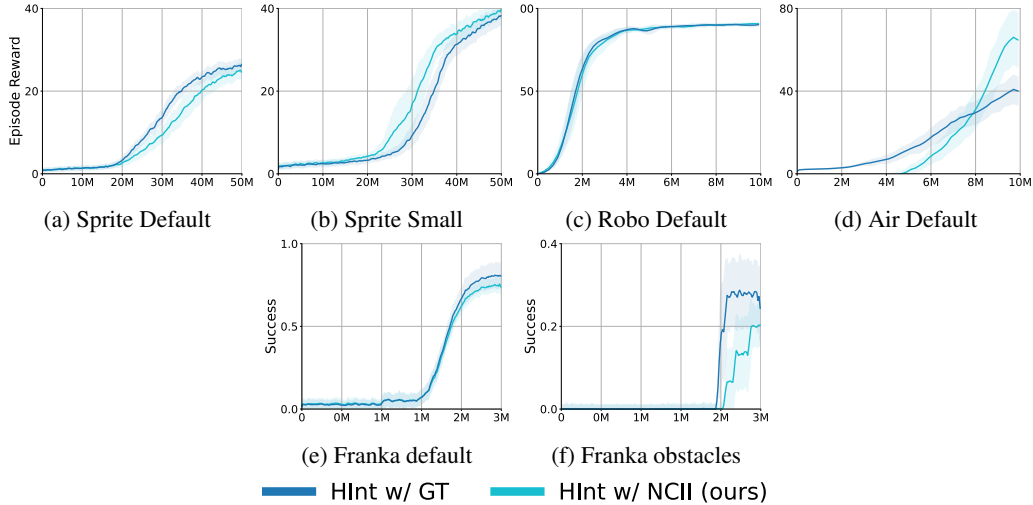


Figure 7: Comparison of HInt and HInt with NCII 5 trials for each. The shading indicates standard error.

D SIMULATED NULLING

Nulling is a powerful tool for lifelong learning settings, where the agent will experience a large number of trajectories in a wide variety of settings. However, many RL domains often only contain a few, specific factors and have a sparse number of interactions. It can be expensive to observe every distribution of possible states in this setting. However, we can often leverage the fact that interactions are rare to identify high-priority states using a passive error, a technique introduced by Chuck et al. (2023).

Intuitively, we can estimate states where there is unlikely to be an interaction affecting the target object by observing if the target object exhibits unexpected behavior. If it *does not*, then there is likely *not* an interaction. For example in an object-based domain, an object is probably *not* interacting

with any other object if it maintains its velocity. More specifically, if an autoregressive model $f : \mathcal{S}_j \rightarrow \mathbb{P}(S'_j | S_j)$ is a good predictor of the next target object state s'_j using only the current target object state s_j , then there is most likely not an interaction. The passive signal is then just the log-likelihood of the observed outcome under the distribution induced by the forward model:

$$e_j^{\text{passive}}(s_j, s'_j) := \log f(s_j)[s'_j] \quad (6)$$

Where $f(s_j)[s'_j]$ is the probability of s'_j under the distribution from $f(s_j)$. When the passive signal is high, this suggests a non-interaction.

We use the passive signal in two ways, first, as a substitute for the null signal. This is because, assuming that there is not any interaction with the target factor in state s , we can null out any other state under the null assumption (if there is no interaction, then the outcome distribution should equal the null distribution). The passive signal can then be used as a proxy for this circumstance since low passive error implies non-interaction. In practice, this means taking the null vector \mathbf{v} , which again is a 0 – 1 vector indicating which objects are present, and randomly zeroing certain indices with some probability. This gives the passive-adjusted null vector $\hat{\mathbf{v}}$:

$$\hat{\mathbf{v}} := \begin{cases} \mathbf{v} \cdot \text{Bernolli}(1 - \epsilon_{\text{sim-null}}) & e_j^{\text{passive}}(s_j, s'_j) > \epsilon_{\text{passive}} \\ \mathbf{v} & \text{otherwise} \end{cases} \quad (7)$$

Where $\epsilon_{\text{sim-null}}$ is the probability of nulling out a value in high passive likelihood states, and $\epsilon_{\text{passive}}$ is the log likelihood threshold to be considered high likelihood. We can then train using NII using $\hat{\mathbf{v}}$ instead of \mathbf{v} .

The second way we use the passive error is to modify the frequency of interaction states observed during training. In many of the domains, interactions are very infrequent, ranging from 0.5% to 0.01%. A model learned with these is likely to simply ignore interactions entirely. Because interactions are low-likelihood, however, they also tend to have low likelihood under the passive model f . Consequently, we upweight low passive likelihood states so that they are sampled on average 20% of the time. This is a technique used in both Chuck et al. (2023; 2024b).

E FILTER CRITERIA

The action path criteria for χ defined in section 4.2 will capture any control exerted by the agent but has three weaknesses. First, because it incorporates information about all edges, it can be susceptible to false positives, where a false positive edge in a different object could induce a path. As an example, imagine the Spriteworld domain with obstacles. If there is a false positive between an object that was controlled and an obstacle that happened to collide with the target object, this would result in a trajectory misclassified as passing the criteria. Second, again because identifying a path requires all edges, this means using the null counterfactual to generate interactions requires an $O(T \cdot n^2)$ operation. Learning these models accurately can be expensive. Finally, while using a binary signal indicates if *any* action influence was exerted, it does not give a good measure of the *amount* of influence. In practice, this is a challenging problem, but we provide a few heuristic strategies in this section.

First, we define 3 simplifying filtering strategies for χ .

1. **Non-Passive:** This is a permissive strategy that rejects any trajectory that does not have a non-trivial interaction, and assigns a random state after the first non-trivial interaction as the hindsight goal. Formally, construct $\mathbb{B}^{(t)}$ using some subset of the interaction graph and assigning the remaining edges to 0. Ignore the edges between (i, i) , and (action, i) (because action edges are typically dense). If there is not a nonzero edge remaining, reject the trajectory. $\iota = \sum_{B^{(t)} \in \mathbb{B}} \sum_{i=0}^n \sum_{j=0}^n \mathbb{B}_{ji}^{(t)} = 0$.

This formulation allows us to utilize a subgraph while still identifying interactions. However, since not all non-passive graphs indicate control, it can be overly permissive.

2. **Non-passive target:** This strategy is a simplification of the general non-passive graph by taking only the row j corresponding to the target object to test for interactions. Formally, for target index j , $\iota := \sum_{B^{(t)} \in \mathbb{B}} \sum_{i=1}^n \mathbb{B}_{ji}^{(t)} = 0$.

Domain	$\epsilon_{\text{null}} = 0.1$	$\epsilon_{\text{null}} = 0.3$	$\epsilon_{\text{null}} = 0.5$	$\epsilon_{\text{null}} = 0.7$	$\epsilon_{\text{null}} = 1.0$	$\epsilon_{\text{null}} = 1.5$	$\epsilon_{\text{null}} = 2.0$	$\epsilon_{\text{null}} = 2.5$	$\epsilon_{\text{null}} = 3.0$	$\epsilon_{\text{null}} = 3.5$
l-in	4.4 ± 3.0	3.4 ± 1.8	3.2 ± 0.5	1.1 ± 1.3	1.2 ± 3.1	1.6 ± 1.3	1.8 ± 0.4	2.2 ± 0.8	2.4 ± 0.9	2.6 ± 2.5
Sprite-1	5.2 ± 2.5	4.9 ± 0.4	2.8 ± 2.5	4.6 ± 3.3	6.4 ± 5.0	21.6 ± 12.5	19.6 ± 4.1	27.4 ± 3.1	39.5 ± 4.7	43.5 ± 17.9

Table 2: Ablation on ϵ_{null} parameter (horizontal) for NCII. Note that because ϵ_{null} indicates a difference in normalized log-likelihood, the parameter is quite robust. Each is run over 3 seeds.

This has the advantage of reducing the computation to a single row of the graph, though it is still overly permissive. However, since a non-passive interaction *must* occur for a control interaction to occur, it will not filter out any interacting trajectories.

3. **Control-target interaction:** This strategy is the most stringent, and identifies a control factor based on the state factor with the most frequent action edge. Call this factor i . Then, the criteria is simply checking if the control factor has an edge with the target factor j , and rejecting trajectories where the control factor does not interact with the target factor.

Formally, $\iota := \sum_{\mathbb{B}^{(t)} \in \mathbb{B}} \mathbb{B}_{ji}^{(t)} = 0$.

This requires, after identifying the control factor, a single interaction test. However, it is stringent, since it is not required that the control factor directly connects to the target object to exert control. In practice, we identify the control factor by learning $2n$ models, one for each state factor, $f^{\text{passive}}(\mathbf{s}_k) \rightarrow \mathbb{P}(S'_k | S_k = \mathbf{s}_k)$ and $f^{\text{action}}(\mathbf{s}_k, \mathbf{a}) \rightarrow \mathbb{P}(S'_k | S_k = \mathbf{s}_k, A = \mathbf{a})$. Then we identify the factor for which adding \mathbf{a} helps the most.

The above strategies can reduce some sensitivity to false positives since they only require inference on a reduced subset of edges. For the same reason, they reduce the computational cost. However, they do not identify the degree of control. We utilize a simple strategy of only keeping a trajectory if the number of interactions according to the testing strategy exceeds a certain count. In other words, there should be at least some amount of interactions if ι is to keep a trajectory. The meaning of frequent is based on an interaction count b , which is defined differently for each method.

1. **Action Graph:** define the interaction count as the number of unique paths:

$$b := \# \text{unique paths from actions to } j$$

. A unique path is any path that contains at least one unique edge.

2. **Non-Passive:** define the interaction count as the number of non-passive edges:

$$b := \sum_{B_{ij}^{(t)} \in \mathbb{B}} \sum_{i=0}^n \sum_{j=0}^n \mathbb{B}_{ji}^{(t)}.$$

3. **Non-Passive Target:** define the interaction count as the number of non-passive edges to the target:

$$b := \sum_{B_{ij}^{(t)} \in \mathbb{B}} \sum_{i=0}^n \mathbb{B}_{ji}^{(t)}.$$

4. **Control-target:** define the interaction count as the number of control-target edges:

$$b := \sum_{B^{(t)} \in \mathbb{B}} \mathbb{B}_{ji}^{(t)}.$$

Then $\iota := b < n_{\text{min interaction number}}$

In practice, we use action-graph interactions in all domains except Spriteworld Obstacles, Robosuite Pushing Obstacles, and Air Hockey Obstacles. In these domains, we use control-target interactions for experimental inference.

F INVESTIGATING ϵ_{NULL}

The ϵ_{null} parameter can have significant effects on the success of the algorithm, since if selected to be too small, this will overestimate the prevalence of interactions, and if too large, interactions will fail

to be identified. While in the experiments for this work we found that a fixed ϵ_{null} is sufficient, as we see in Table 2, we provide the following strategies for identifying this parameter using information from learning the null model.

First, learn a null forward model $f : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \rightarrow P(\cdot|S, A)$ that uses information about all input states to predict a distribution over S'_i , as in Section 4. Then compute the difference in value from the null operation on the dataset:

$$\text{diff}(\mathbf{s}, \mathbf{a}, \mathbf{s}_j, \theta)_{\text{ji}} = \log f(\mathbf{s}, \mathbf{a}, \mathbb{B}(\mathbf{v}); \theta)_j[\mathbf{s}_j] - f(\mathbf{s}, \mathbf{a}, \mathbb{B}(\mathbf{v}) \circ S_i; \theta)_j[\mathbf{s}_j] \quad (8)$$

Now, our objective is to identify the differences that most likely correspond to interactions. Non-interactions will generally have low error, since nulling should have no effect on the outcome $\text{diff}(\mathbf{s}, \mathbf{a}, \mathbf{s}_j, \theta)$. Thus, there should be a cluster of low likelihood difference outcomes. Since interactions mean that the non-nulled evaluation will be higher likelihood, $\text{diff}()$ will be higher for these values, corresponding to the second cluster. In practice, we can apply a 2-mode clustering algorithm to get the interaction and non-interaction clusters, and take the midpoint (or some other in-between point) of the mean of the two clusters.

G NETWORK ARCHITECTURES

In this section, we describe the network architectures used in this work. The same architecture is used for h, f , the interaction and forward dynamics networks.

G.1 POINTNET ARCHITECTURES

Pointnet (Qi et al., 2017) utilizes a 1D convolution-based architecture and an order-invariance reduce function. In this work, we utilize a multilayer pointnet, which can re-append an embedding output by the final layer as an input. Our Pointnet uses a 1D convolution to embed the states and action $\mathbf{s}_1, \dots, \mathbf{s}_n, \mathbf{a}$, masks them with \mathbf{v} , and then follows that with a second 1D convolution. This architecture is visualized in Figure 8.

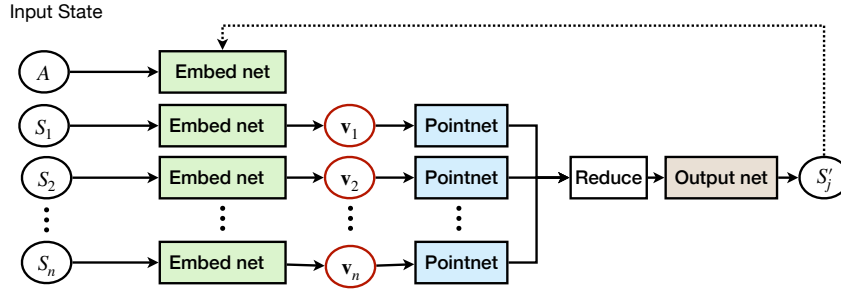


Figure 8: The Pointnet-based architecture used for interactions. Multilayer methods repeatedly re-append the output to the embeddings. Shared colors (green, yellow) denote weight sharing through 1D convolution.

G.2 GRAPH ARCHITECTURES

We use graph neural networks (GNNs) to model the dynamics, while 1D convolutions are applied to embed the state and actions. For message passing on GNN, we utilize GCNConv layers (Kipf & Welling, 2017), and when nulling out objects, we directly remove the corresponding nodes and its edges from the graph. The architecture is visualized in Figure 9.

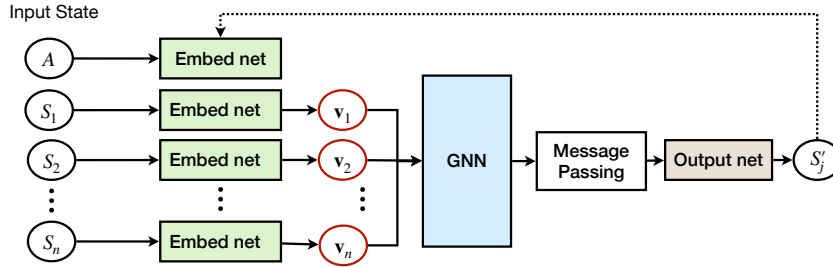


Figure 9: The GNN architecture used for interactions. Similarly, shared colors (green, yellow) denote weight sharing through 1D convolution.

G.3 TRANSFORMER ARCHITECTURES

To adapt to the transformer backbone (Vaswani, 2017), we model the dynamics using a transformer and nullify the effects of objects by setting their interactions to zero in the cross-attention maps. However, a challenge occurs when using multiple cross-attention layers: nulling out attention in one layer does not prevent information exchange in subsequent layers. Results from Random DAG domains (Table 3) also indicate that transformers perform worse than PointNet or GNN, which we will address in future work.

Method	NCII w/ Point	NCII w/ Graph	NCII w/ Transformer
1-in	0.8 ± 0.2	1.0 ± 0.1	2.6 ± 0.3
2-in	1.4 ± 0.2	1.4 ± 0.2	4.9 ± 1.7
6-in	25.1 ± 5.8	26.3 ± 5.2	41.3 ± 8.2

Table 3: Accuracy of inference in evaluated domains from states using NCII with different backbones. Interactions are reweighted to be 50% of the test dataset.

H ENVIRONMENT DETAILS

H.1 SPRITEWORLD

The three 2D Spriteworld domains consist of a target ball being pushed to a goal region by a control ball in a low friction, gravity-free 7x7 meter region. The action space consists of forces upon the control ball, and the goal space is a 1m diameter circle. The reward function is sparse, with 0 reward outside and 1 reward inside the goal region. Because controlling the control ball is already challenging, and when struck the target ball is likely to pass through rather than stay in the goal region, this task can take a very substantial number of interactions to train. In the variants, we add 2 obstacles of similar mass to the target ball which impede it from reaching the goal. Ground truth interactions in this domain consist of contacts between the different objects.

The three variants differ according to the radius of the goal and target objects, and the presence of obstacles. In the Default domain, the radius of the control and target objects are 0.5 meters, while in the **small** and **Obstacles** domains they are 0.3 meters. In the obstacles domain are two additional obstacles, a triangle and a circle, of similar mass to the target object and where the radius of the polygon (distance from centroid to further vertex) and circle is 0.5 meters. Each domain uses 100 time steps for a trajectory before timing out.

H.2 ROBOSUITE

The three Robosuite domains involve pushing a block to a desired target location using end effector control of a PANDAs robotic arm. The workspace is a $0.6 \times 0.6 \times 0.3$ meter region in length, width height respectively. The action space consists of desired end effector deltas, using an OSC controller to achieve the desired delta position. The goal space is a 0.05m diameter circle. Again, the reward function is sparse, with 0 reward outside and 1 reward inside the goal region. Increasing the dimensionality by moving the gripper in 3D space means that interactions are even less frequent,

even though the task is over a much smaller area. In the **obstacle** variant we add 2 immovable 0.05m blocks to the domain, initialized randomly such that they do not lie on top of the goal. Once again, Ground truth interactions in this domain consist of contacts between the different objects.

The three variants differ according to the size of the target block, and the presence of obstacles. In the **Default** domain, the cube side length of the target block is 0.015 meters, while in the **small** and **obstacles** domain it is 0.007 meters, where the obstacle domain has the two additional 0.05m obstacles. Each domain uses 100 time steps for a trajectory before timing out.

H.3 AIR HOCKEY

The three 2D Air Hockey domains consist of a puck being struck into a goal region using a paddle in a low friction 2x1 meter region, where there is gravity pulling the puck down towards the paddle. The action space consists of forces upon the paddle, and the goal space is a 0.2m diameter circle. In this domain only, we use a shaped reward instead of a sparse one, although hindsight still proves to be useful because of the complexity of the dynamics. The reward function is densified with a 12 distance, which was necessary for any policy to learn in this domain. The agent receives $-\frac{1}{2}\|s_{\text{puck}} - g\|_2^2$ reward outside and 1 reward inside the goal region. The challenge in this domain is to use a sparse interaction to achieve a downstream effect, with the puck constantly moving in and out of the goal. In the variants, we add 2 blocks of high mass to the target ball which can impede the puck from reaching the goal. Ground truth interactions in this domain consist of contacts between the different objects.

The three variants differ according to the radius of the puck and paddle, and the presence of obstacles. In the **Default** domain, the radius of the puck is 0.03m and puck is 0.05 meters, while in the **small** and **obstacle** domain the puck is 0.02 meters and the puck is 0.03 meters. The obstacle domain additionally adds three random obstacles. The goal in all domains is 0.15m radius. Each domain uses 100 time steps for a trajectory before timing out.

H.4 FRANKA KITCHEN

In the Franka Kitchen domain, a robot with 9 degrees of freedom is tasked with controlling various kitchen objects, including the top and bottom burners, light switches, sliding and hinged cabinets, kettle, and microwave. Each object corresponds to a specific sub-task, and the robot must perform a sequence of tasks, where each is associated with the goal positions of its joints. The sparse reward is computed based on the number of completed tasks. Here in the **Default** domain, the robot’s task is to open the microwave (three objects: desk and microwave). In the **Obstacle** domain, the robot’s task is to open the microwave and the sliding cabinet, where the additional objects such as the kettle and burner switches are added as obstacles.

I TRAINING DETAILS

In this section we describe the hyperparameters and training details for NCII and HInt.

All null experiments were collected with 10 seeds between 0-9. All RL experiments used 5 seeds between 0-4. The experiments were conducted on machines of the following configurations:

- 4×Nvidia A40 GPU; 8×Intel(R) Xeon(R) Gold 6342 CPU @2.80GHz
- 4×Quadro RTX 6000 GPU; 4×Intel(R) Xeon(R) Gold 6342 CPU @2.80GHz
- 4×Nvidia 4090 GPU; 8×Intel(R) Xeon(R) Gold 6342 CPU @2.80GHz
- 2×Nvidia A100 GPU; 8×Intel(R) Xeon(R) Gold 6342 CPU @2.80GHz

Encoding Dim	512
Hidden	3×512
Activation	Leaky ReLu

Table 4: Forward/inference Model

Parameter	Value
ϵ_{null}	1 (log-likelihood space)
Minimum Normalized distribution variance	0.001
Distribution	Diagonal Gaussian
Learning Rate	1×10^{-4}

Table 5: Null Parameters

Parameter	Value
Algorithm	DDPG
Batch Size	1024
Optimizer	Adam
Actor/critic learning rate	1×10^{-4}
Exploration Noise	0.1
γ	0.9
Hidden Layers	2×512
τ	0.005

Table 6: Reinforcement Learning Parameters

Domain	Timeout	Normalized Goal Epsilon	Null Train Steps	RL Train Steps
Spriteworld Default	100	0.1	1M	50M
Spriteworld Small	100	0.15	1M	50M
Spriteworld Obstacles	100	0.2	-	50M
Spriteworld Velocity	100	0.2	-	50M
Robosuite default	100	0.15	1M	10M
Robosuite small	100	0.15	-	10M
Robosuite obstacles	100	0.15	-	15M
Air Hockey default	400	0.2	1M	10M
Air Hockey small	400	0.2	-	20M
Air Hockey obstacles	400	0.2	-	20M
Franka Kitchen default	200	0.2	2M	3M

Table 7: Domain Specific Parameters

Domain	HInt learned	HInt	Hind	Prio	FPG	Vanilla	ELDEN	CAI
Sprites (50M)	62.21	22.10	23.27	26.02	17.15	22.57	40.12	69.34
Air (10M)	12.50	4.33	4.59	6.05	5.24	3.31	9.75	17.23
Robo (10M)	21.17	11.52	15.32	11.28	8.59	8.21	18.28	25.20
Franka (3M)	16.68	9.26	8.45	8.92	10.39	9.16	13.25	22.48

Table 8: Wall Clock Compute Time in Hours

J HINDSIGHT SAMPLING ABLATION

In this work, we focused on the introduction of interactions into Hindsight Filtering using the “final” sampling scheme, which takes the last state of a trajectory as the hindsight goal. In practice, different sampling schemes can be used for hindsight [Andrychowicz et al. \(2017\)](#), including sampling any state after the one observed “future” or any state from the trajectory “episode.” We provide some analysis

Method	NCII w/ Point	JACI	Gradient	Attention	NCD
1-in-nonlinear	0.9 ± 0.2	2.4 ± 0.8	32.5 ± 6.1	37.4 ± 0.7	21.2 ± 1.1
2-in-nonlinear	2.3 ± 0.1	2.5 ± 0.2	36.4 ± 0.4	21.8 ± 0.9	19.8 ± 2.0
40-dim	1.4 ± 0.1	2.4 ± 0.5	34.7 ± 4.4	26.4 ± 6.9	12.5 ± 0.8

Table 9: Misprediction rate (lower is better) of inference in additional domains from state, similar to Table 1. Interactions are reweighted to be 50% of the test dataset. Boldface indicates within ~ 1 combined standard deviation of the best result. k-in-nonlinear uses nonlinearities in the random DAG instead of a linear relationship. 40-dimensional uses a 40 dimensional state.

suggesting that hindsight filtering is applicable in any sampling scheme in Figure 10. Note again that in general, we used the “final” sampling strategy for all other experiments (Figure 4, Figure 7, Figure 12).

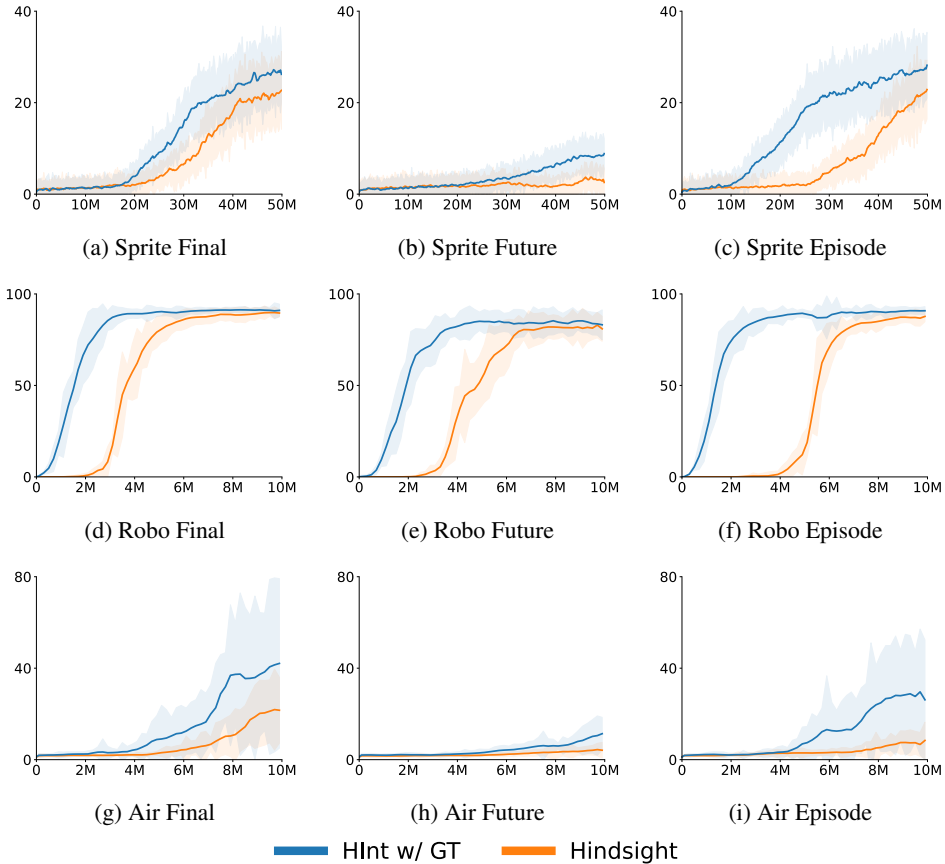


Figure 10: Comparison of HInt and Hindsight using different HER sampling schemes “final”, “future” and “episode”, with 3 trials for each sampling scheme. In these experiments, we modified the sampling scheme for both HInt and Hindsight to the sampling strategy indicated in the caption. The shading indicates standard error. Sprite-, Air- and Robo- domains each use the default variant. The Y axis is average reward per episode.

K VISION EXPERIMENTS

While the objective of this work is to demonstrate a generally applicable method for utilizing counterfactual nulls for interaction inference (NCII) and interaction filtering for GCRL (HInt), we briefly explore the scaling capabilities of NCII and HInt. In both interaction inference and GCRL, performance in higher dimensional states remains a challenging problem. While our results certainly

Method	NCII w/ Point	JACI	Gradient	Attention	NCD
Sprite-1-vision	13.9 \pm 0.5	18.1 \pm 0.2	26.4 \pm 2	39.6 \pm 4.5	21.2 \pm 1.1

Table 10: Misprediction rate (lower is better) of inference in Spriteworld-1 domain. Interactions are reweighted to be 50% of the test dataset. Boldface indicates within ~ 1 combined standard deviation of the best result.

suggest that the NCII is competitive with baselines in both domains, these scaling questions remain unsolved.

Empirically, we take a segmentation of each object in the frame (given from the simulator) in the Sprite default domain, and train a variational autoencoder Kingma (2013) on a frame stack of 3 frames of each segmented object with a latent dimension of 128. We then append object-centric features pixel position and velocity, duplicated to 128 dimensions. This is then used as the input for both NCII and RL using HInt. We first compare NCII to interaction inference baselines in the Spriteworld Default domain in Table 10. Then we provide a performance curve for HInt when compared against the Hindsight and Vanilla RL baselines in Figure 11.

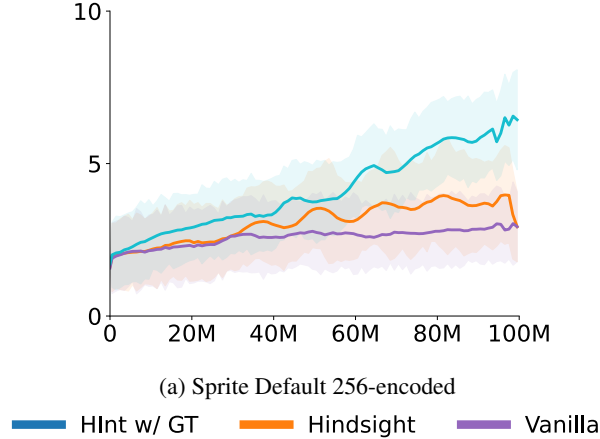


Figure 11: Comparison of HInt, Hindsight and Vanilla RL, 3 trials for each, on Spriteworld Default using 256-dimension image encodings. The shading indicates standard error. The Y-axis is average reward per episode.

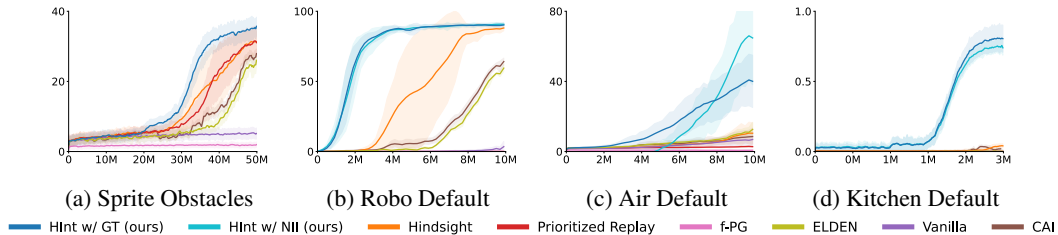


Figure 12: Addition of Causal Action Influence (CAI) Seitzer et al. (2021) baseline in selected domains. This baseline is similar to ELDEN, but has more inductive bias towards actions. The Y axis is the average reward per episode.

L ADDITIONAL COMPARISONS

We also compare with **Causal action influence (CAI)** (Seitzer et al., 2021). CAI uses conditional mutual information to infer the local causal relations, detecting when and what the agent can influence the state variables with its actions. Then they employ the influence as the intrinsic motivation for

exploration that benefits sample efficiency. The results in selected domains are given in Figure 12, where CAI performs similarly to ELDEN empirically.

M ROLLOUT VISUALIZATIONS

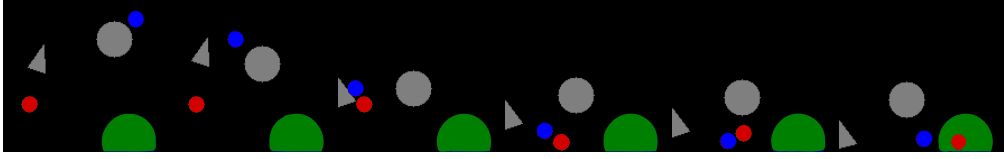


Figure 13: Selected frames from a successful policy rollout for Spriteworld Obstacles

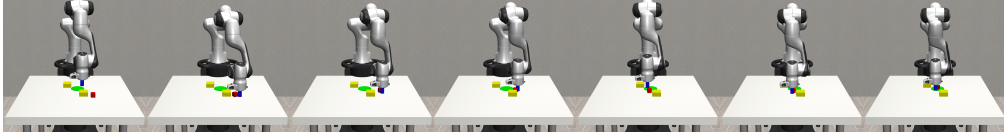


Figure 14: Selected frames from a successful policy rollout for Robosuite Obstacles

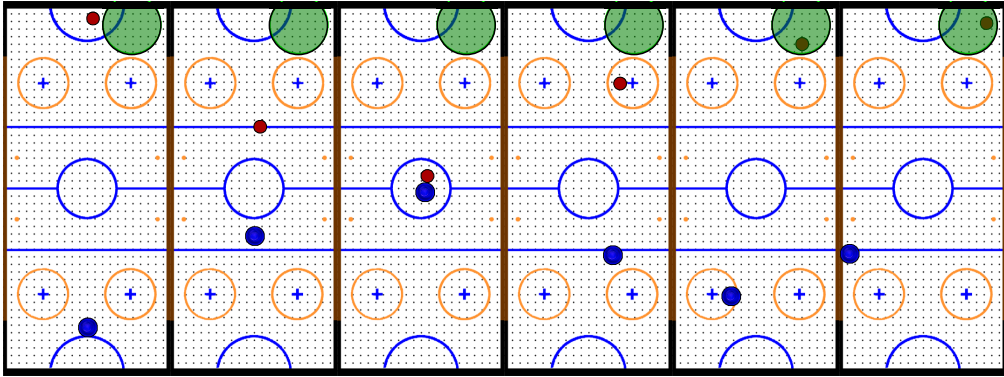


Figure 15: Selected frames from a successful policy rollout for Air Hockey Default

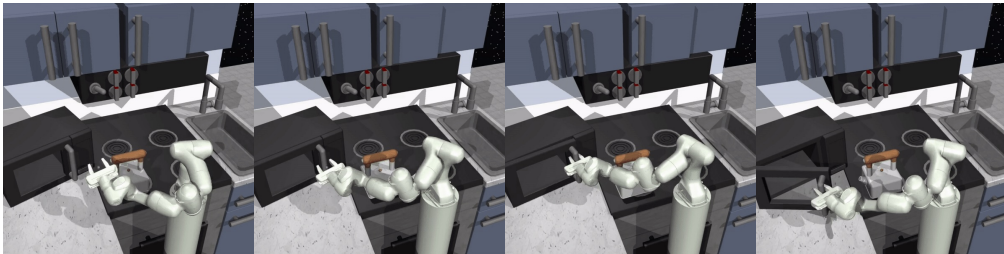


Figure 16: Selected frames from a successful policy rollout for Franka Kitchen