

## A Model details

In a standard fully-connected multilayer perceptron (MLP), the input is treated as a whole and input into the first linear layer. It blends all information into the feature of subsequent layers, making it difficult to separate the cause and effects. To highlight the difference between our model and traditional MLP, we plot the detailed model architecture of the COMPASS in Figure. 6.

More specifically, we design an encoder-decoder structure in  $f$ , with  $\mathcal{G}$  as a linear transformation applied to the intermediate features. First, the encoder operates on each dimension of  $\epsilon$  independently to generate features  $z_\epsilon \in \mathbb{R}^{|\mathcal{E}| \times d_z}$ . Then the causal graph is multiplied by the features to generate the inputs for the decoder, i.e.,  $g_\epsilon = z_\epsilon^T \mathcal{G} \in \mathbb{R}^{d_z \times K}$ , where  $d_z$  is the dimension of the feature. Similarly, the action sequence  $\mathbf{a}$  is passed through the encoder and transformation to produce the feature of the action sequence  $g_a \in \mathbb{R}^{d_z \times K}$ . Finally,  $g_\epsilon + g_a$  is passed through the decoder to output the prediction  $\hat{d}_T$ .

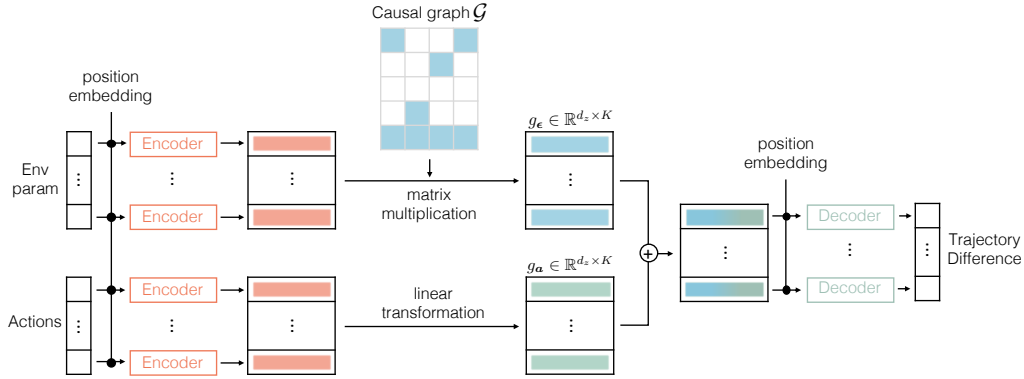


Figure 6: COMPASS model architecture.

## B Experimental Details

### B.1 Experimental Configurations for Robot Simulation Setup

A simulator was developed to mimic a simulated air hockey table with dimensions matching the real-world setup using robosuite [13]. This allowed us to gather rollout trajectories and training data within the simulated environment. The specific dimensions of the objects within the simulation, such as the table size, are detailed in Table 3. Additionally, for the sim-to-sim experiment configuration, default simulation parameters and the ground-truth simulated real environment parameters are presented in Table 4.

**State and action space.** The observation space for the RL agent comprises 6 dimensions. It includes the initial position of puck1 (3D), and the initial position of puck2 (3D). The RL agent’s action space consists of 4 dimensions: the initial position of the pusher in the x-direction, the initial position of the pusher in the y-direction, the shooting angle of the pusher, and the pushing velocity of the pusher. We fixed the initial position of puck1 and puck2. Note that the shooting angle is relative to the line connecting the center of the pusher and puck1.

**Reward function.** The reward is calculated as  $-10 \times \|[x_{puck1}, y_{puck1}, z_{puck1}] - [x_{goal}, y_{goal}, z_{goal}]\|_2$ . To provide additional incentive for reaching the goal, the distance penalty term is divided by 2. Furthermore, a terminal reward is given if the hockey stays within the success region at the last time step. It is important to mention that the reward is not accumulated throughout the horizon. Instead, it only considers the final Euclidean distance between puck2 and the goal center. For further numerical details and specifications, please refer to Table 5.

Table 3: Mujoco Simulation Environment Setup

	X (m)	Y (m)	Z (m)	Radius(m)
Air hockey table	0.0	0.0	0.8	[0.45, 0.9, 0.035]
Puck1	-0.15	0.0	0.8	0.0255
Puck2	-0.075	-0.075	0.8	0.0255
Goal point	0.43	0.0	0.8	0.15
Obstacle bar	0.1	0.0	0.8	[0.025, 0.18, 0.025]

Table 4: Sim-to-Sim Env Parameters

Env param	Default Simulation Env Parameters	Simulated “Real” Env Parameters
pusher@actuation@vel_discount	0.75	0.85
pusher@dyna@damping	-10.0	-6.0
puck1@dyna@damping	-10.0	-6.0
puck1@dyna@friction_sliding	0.05	0.04
puck2@dyna@damping	-10.0	-6.0
puck2@dyna@friction_sliding	0.05	0.03
front_wall, back_wall, left_wall, right_wall, obstacle@dyna@damping	-10.0	-6.0
env@camera@bias_x	0.0	+0.03
env@camera@bias_y	0.0	-0.02

## B.2 Experimental Configurations for Real Robot Setup

This is a top-down view of the mini air hockey table we used to collect real trajectories. The dimensional attributes of each component are annotated in Figure 7, while green dashed lines distinctly demarcate the designated goal area. We used Kinova Gen 3 robot platform and ROS [43] to interface with it. We installed a top-down Intel RealSense D345f RGB camera to track the position of puck1 and puck2.

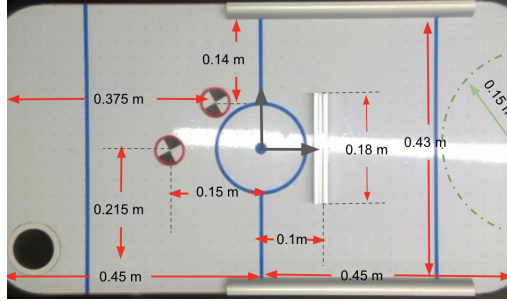


Figure 7: Top-Down View of Table Air Hockey.

## C Implementation Details

We reproduced the baseline implementation EXI-Net [26], NPDR [15] and Tune-Net [24] based on the papers and released code base. We utilized software packages such as PyTorch [44], sbi [45], StableBaseline3 [46], and OpenCV [47]. We report the hyperparameters used for all algorithms,

Table 5: Simulation Environment Setup

Simulation Parameters	
Action space	low: [-0.24, 0.065, -0.157, 0.3] high: [-0.21, 0.085, 0.157, 0.5]
Observation space	low: [-inf] — high:[inf]
Terminal reward	2.25
Simulation horizon	50 time steps
Simulation timestep	0.05 s

Table 6: Soft Actor-Critic Hyperparameters

Parameters Name	Values
learning_rate	3e-4
gradient_steps	32
batch_size	32
train_freq	8
ent_coef	0.005
net_arch	[32, 32]
policy	“MlpPolicy”
env_number	64
buffer_size	1,000,000
learning_starts	100
tau	0.005
gamma	0.99
action_noise	None
stats_window_size	100

including the proposed COMPASS method, in Table 7, 8, 9, and 10, which provide a comprehensive list of the parameters we utilized to reproduce the results.

**Soft Actor-Critic hyperparameters.** We use SAC implementation in StableBaseline3 [46] to train the RL agents. The training hyperparameter is shown in Table 6.

Table 7: COMPASS hyperparameters

Description	value	variable_name
Shared Hyperparameters		
Number of iterations	10	n_round
Retrain in each iteration (if False, keep using the model trained in the first iteration)	True	retrain_from_scratch
Number of rollouts in each iteration	640	n_samples_per_round
Number of command actions in each iteration	10	n_cmd_action
Number of epochs	4000	n_epochs
Batch size	64	batch_size
Learning rate	0.001	learning_rate
Algorithm-Specific Hyperparameters		
Network encoder dimension	32	emb_dim
Network hidden dimension	[256, 256]	hidden_dim
Causal dimension	32	causal_dim
Sparsity weight of the loss function	0.003	sparse_weight
Sparsity weight discount	0.5	sw_discount
Loss function	MSE + Sparsity	loss_function
Optimizer	Adam	optimizer

Table 8: EXI-Net hyperparameters

Description	value	variable_name
Shared Hyperparameters		
Number of iterations	10	n_round
Retrain in each iteration (if False, keep using the model trained in the first iteration)	True	retrain_from_scratch
Number of rollouts in each iteration	640	n_samples_per_round
Number of command actions in each iteration	10	n_cmd_action
Number of epochs	4000	n_epochs
Batch size	64	batch_size
Learning rate	0.001	learning_rate
Algorithm-Specific Hyperparameters		
Network hidden dimension	[256, 256]	hidden_dim
Loss function	MSE	loss_function
Optimizer	Adam	optimizer

Table 9: NPDR hyperparameters

Description	value	variable_name
Shared Hyperparameters		
Number of iterations	10	n_round
Retrain in each iteration (if False, keep using the model trained in the first iteration)	True	retrain_from_scratch
Number of rollouts in each iteration	640	n_samples_per_round
Number of command actions in each iteration	10	n_cmd_action
Algorithm-Specific Hyperparameters		
Prior distribution type	Uniform	prior
Inference model type	maf	inf_model
Embedding net type	LSTM	embedding_struct
Embedding downsampling factor	2	downsampling_factor
Posterior hidden features	100	hidden_features
Posterior number of transforms	10	num_transforms
Normalize posterior	False	normalize_posterior
Density estimator training epochs	50	num_epochs
Density estimator training rate	3e-4	learning_rate
Early stop epochs once posterior converge	20	stop_after_epochs
Use combined loss for posterior training	True	use_combined_loss
Discard prior samples	False	discard_prior_samples
Sampling method	MCMC	sample_with
MCMC thinning factor	2	thin

Table 10: Tune-Net hyperparameters

Description	value	variable_name
Shared Hyperparameters		
Number of iterations	1	n_round
Retrain in each iteration (if False, keep using the model trained in the first iteration)	False	retrain_from_scratch
Number of rollouts in each iteration	6400	n_samples_per_round
Number of command actions in each iteration	10	n_cmd_action
Number of epochs	4000	n_epochs
Batch size	64	batch_size
Learning rate	0.001	learning_rate
Algorithm-Specific Hyperparameters		
Network input dimension (Pair of Trajectory and Action dimension)	(2, 304)	(dim_pair, dim_state)
Network output dimension (Tunable env param dimension)	64	dim_zeta
Env param update iteration	10	K
Network hidden dimension	[256, 256]	hidden_dim
Loss function	MSE	loss_fn
Optimizer	Adam	optimizer