PhyPlan: Learning To Plan Tasks with Generalizable and Rapid Physical Reasoning for Embodied Manipulation

Ankit Kanwar*, Hartej Soin*, Abhinav Barnawal*, Mudit Chopra, Harshil Vagadia, Tamajit Banerjee, Shreshth Tuli, Rohan Paul and Souvik Chakraborty

{Dept. of Comp. Sci. & Engg., Yardi School of AI, Dept. of Applied Mech.}, Indian Institute of Technology Delhi, India
{ankitkanwar2003, htsoin, abhinav.barnawal.iitd, muditchopra123, harshilvagadia11, tamajitbuba3, shreshthtuli}@gmail.com, rohan@cse.iitd.ac.in, souvik@am.iitd.ac.in

Abstract

Given the task of landing a ball in a goal region beyond direct reach, humans can often throw, slide, or rebound objects against the wall to attain the goal. Enabling robots to replicate such reasoning is non-trivial as it requires multi-step planning and involves a mixture of discrete and continuous action spaces, a sparse and sensitive reward structure, computationally expensive simulations, and an incomplete understanding of the environment's physics. We present PhyPlan, a physics-informed and adaptable planning framework for efficient multi-step physical reasoning. At its core, PhyPlan comprises of Generative Flow Networks (GFlowNets) and Monte Carlo Tree Search (MCTS) to explore and evaluate sequences of object interactions. GFlowNets sample discrete action sequences in proportion to their associated reward, enabling broad and reward-driven exploration of the discrete planning space. MCTS complements this by adaptively balancing the use of a fast but approximate pre-trained physics-informed dynamics predictor and costly but accurate environment rollouts, ensuring both speed and precision in planning. The known and actual physics discrepancy is captured using Gaussian Process Regression. Experiments on benchmark simulated tasks requiring composition of collisions, slides, and rebounds demonstrate that PhyPlan achieves a 45% higher success rate and up to 3× efficiency gains over state-of-the-art model-based reinforcement learning approaches.

Website — https://phyplan.github.io

1 Introduction

Consider a scenario where a robot is tasked with placing a ball inside a directly unreachable empty box via sequential tool manipulation. Such problems require multi-step planning involving (i) a combination of discrete and continuous action spaces, (ii) a sparse and sensitive reward structure — the region of high rewards is concentrated around the goal and minor perturbations in actions lead to hard-to-predict changes in rewards, and (iii) an incomplete understanding of the environment's dynamics. Humans are naturally adept at navigating around such problems, with a rich body of research crediting an intuitive physics engine for perceptual goal-directed reasoning (Davis 2008; Bear et al. 2021). However, the lack of equivalent physical priors in robots makes the problem especially difficult, requiring computationally expensive demonstrations to overcome.

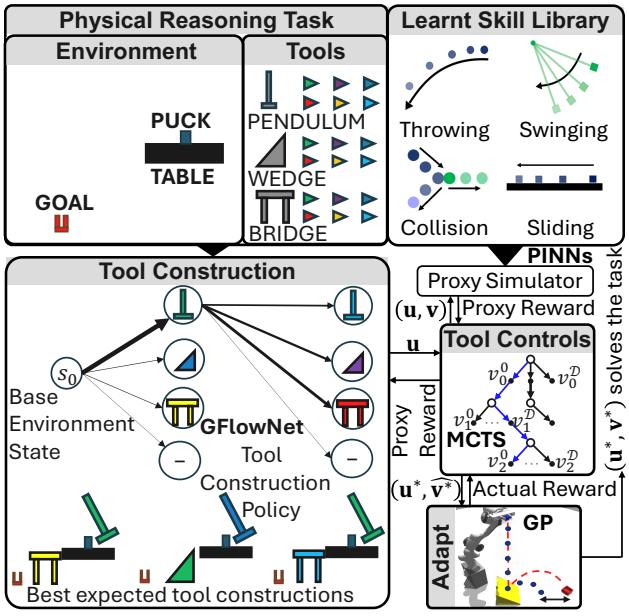


Figure 1: Physical reasoning tasks involve determining which object interactions (hitting, sliding, falling etc.) can be sequenced and their control parameters (e.g., height of falling, extent of swing etc.). Our approach couples (i) GFlowNets for postulating tool constructions (ii) MCTS with physics-aware dynamics predictors for rapid prediction of sequential tool interactions, without expensive simulations, and (iii) a $\mathcal{GP}$ to learn the model discrepancy between the proxy and the actual environment reward.

Recent works by Allen, Smith, and Tenenbaum (2020) solve physical tasks via physics simulators, while Bakhtin et al. (2019) benchmark model-free methods. These approaches are computationally expensive, data-hungry, and require training from scratch for each new task. Toussaint et al. (2019) observe that physical reasoning tasks are a se-

*These authors contributed equally.

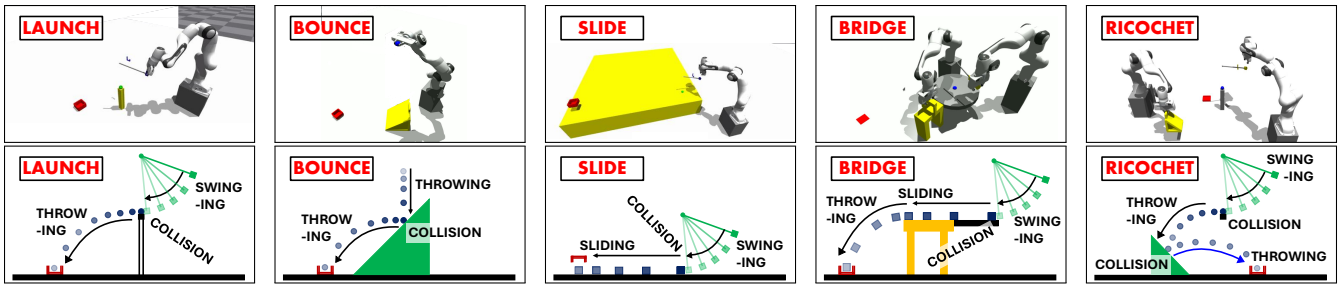


Figure 2: Benchmark Physical Reasoning Tasks: This paper considers five tasks requiring the robot to place a ball/puck in a goal (red box) outside its direct kinematic range. Hence, the robot must use sequential dynamic interactions such as collision, sliding, etc., to land the ball in the goal. Further, it must infer unknown physical parameters such as coefficient of friction to generalize to novel task settings.

quence of elementary dynamic interactions underscoring the need for "physical priors". Consequently, they improve scalability with symbolic abstractions of physical priors, which can be embedded into a planning domain for Planning Domain Definition Language (PDDL)-style planning to arrive at multi-step plans. However, their work faces brittleness due to modeling errors.

In response to the above challenges, this paper introduces an adaptable physics-informed planning framework, PhyPlan, for multi-step physical reasoning. PhyPlan learns structured dynamics predictors by incorporating coarse physics models and Parametric Physics-Informed Neural Networks (PINNs) (Raissi, Perdikaris, and Karniadakis 2019; Chakraborty 2020) that predict the effect of composing cascaded object interactions (e.g., a ball sliding, rebounding, and falling). Physical reasoning is performed using a novel composition of Generative Flow Network (GFlowNet) (Bengio et al. 2023) and Monte Carlo Tree Search (MCTS) that uses physics-informed dynamics predictors to explore object interactions and controls towards the intended goal. This novel composition of GFlowNet and MCTS allows PhyPlan to efficiently handle the mixed action space involving both discrete and continuous variables. For rapid reasoning, we propose and exploit a soft-adaptive MCTS. The known and actual physics discrepancy is captured using Gaussian Process Regression. Results show that PhyPlan achieves efficient task resolution with minimal reliance on real-world executions outperforming the state-of-the-art methods, such as Model-Based Policy Optimization, while consuming significantly less time. Overall, PhyPlan excels in (i) data efficiency, (ii) reduced planning time, and (iii) generalizability across novel physical reasoning tasks.

2 Related Works

Endowing AI agents with physical reasoning abilities has received attention in AI and cognitive sciences (Lake et al. 2017; Schwettmann et al. 2018). Authors have proposed architectures that model intuitive notions of physical stability (Hamrick et al. 2018), dynamic sliding (Wu et al. 2015) and other non-rigid interactions (Allen et al. 2022). Allen, Smith, and Tenenbaum (2020) present a simulate-correct approach to learn compositional tool use. Others propose benchmark tasks (Bakhtin et al. 2019; Xue et al. 2023) and demonstrate the limited ability of contemporary approaches in tasks involving multi-step, continuous dynamic interactions. While prior works focus on simplistic 2D tasks, this work considers challenging multi-step 3D tasks (with an explicit agent) modeling both selection and control of tools.

Researchers in the robotics community have focused on imparting "skills" such as grasping, pushing etc. via demonstration (Chen et al. 2022; Park et al. 2020; Finn and Levine 2017), policy optimization Zeng et al. (2020), active exploration (Moses et al. 2020) or learning partially-parameterized skills (Ijspeert et al. 2013). However, these efforts focus on short horizon skills delegating reasoning over skill sequences to a decoupled planning process. Works focused on *task and motion planning* (Toussaint et al. 2019) impose symbolic PDDL-style symbolic abstractions over continuous skills but assume that the skills are already trained with explicit pre-/post-conditions. In contrast, this work incorporates both learning primitive skill as well as composing them to solve a task. PDDL+ (Fox and Long 2006) extends PDDL-style planners and incorporate continuous variables. PDDL+ requires a symbolic model of actions with deterministic post-conditions. Our work instead learns a physics-informed action models and can reason with uncertainty in action outcomes.

Recent works leverage the zero-shot capacity of large language models to obtain a plan (Gao et al. 2024). However, their general ability to reason over continuous physical interactions is limited and the same is corroborated in our experiments. Others attempt to combine LLMs with a simulator for iterative feedback (Hao et al. 2023). However, the need for high-fidelity dynamic simulation (NVIDIA 2022; Todorov, Erez, and Tassa 2012) involved in physical reasoning tasks yield such approaches ineffective in large exploration spaces. Instead of decoupled iterative feedback, our approach learns fast physics-aware simulators (Raissi, Perdikaris, and Karniadakis 2019) which is tightly integrated with online planning via model-based RL techniques.

3 Preliminaries

The robot's environment is composed of a set of objects and a discrete set of regions where objects can be placed. The

robot observes its workspace via a visual and depth sensor and is assumed equipped with a perception system yielding the object pose and the metric model for each object in the scene. The robot is assumed equipped with primitive skills to grasp, transport and release objects at a specific location. The agent's goal is to determine a solution for a physical reasoning task that involves deciding an arrangement of a set of objects (or "tools") such that when an interacting object (such as a ball) interacts dynamically with the scene (i.e., undergoing collisions, bounce, slide etc.) the object reaches a designated goal region.

Formally, we define the agent's world representation as follows. Let S denote the set of environment states. Let $\mathcal{X} \subseteq S$ denote the set of terminal states where all objects dynamically stable (e.g., when the ball comes to a halt on a supporting surface like a table or on the ground). Let $\mathcal{G} \subseteq \mathcal{X}$ denote the goal states (e.g., a region defined by a container into which a ball must fall to complete the task). Positive reward is assumed associated with each terminal state $\mathbf{s} \in \mathcal{X}$ and initial state $\mathbf{s_0} \in S$ as $\mathcal{R}(\mathbf{s}) = 1 - \frac{\min_{\mathbf{g}\in\mathcal{G}} d(\mathbf{s},\mathbf{g})}{\min_{\mathbf{g}\in\mathcal{G}} d(\mathbf{s_0},\mathbf{g})}$ where $d(.)$ is the standard euclidean distance function. Let $o \in \mathcal{O}$ denote the set of objects in the scene and let $\mathcal{W} = \{o_1, o_2, \ldots, o_H\}$ denote the specific set of H objects present in the robot's environment.

Next, we define the object interactions that the agent reasons with while reasoning about a physical reasoning task. Let $K(o)$ denote the *discrete* set of variants for each object (e.g., discrete set of frictional/restitution coefficients) and let $L(o)$ denote the set of discrete world regions where the object can be placed[1]. Note that the assignment of objects to regions implicitly represents the tool sequence for a set of objects (e.g., tools such as pendulum, table or wedges can be arranged in different orders and would hence lead to different dynamic simulations of a falling ball). Let the set $\mathcal{U}(o) = K(o) \times L(o)$ expresses the combinatorial space of *discrete* variables associated with each object, where each instance $u_o \in \mathcal{U}(o)$ defines a tool construction / combination (tool variants and their regions where the tools are placed).

Further, we model the continuous configuration space of any object, e.g., the release angular orientation of a pendulum, the angle of a pendulum used for hitting or the relative distance between a bridge from a sliding surface. Formally, let $\mathcal{V}(o)$ define the set of *continuous* control variables associated with each object. Note that modulating each variable $v_o \in \mathcal{V}(o)$ impacts future dynamic interactions (e.g., releasing a pendulum from an exaggerated angle can impact greater momentum to a ball and hence can slide it for a longer distance). Finally, an action of the robot $\boldsymbol{a} \in \mathcal{A}$, the action space, can be represented as the set $\boldsymbol{a} = (\boldsymbol{u}, \boldsymbol{v}) = \{u_1, v_1, \ldots, u_H, v_H\}$ for a world with H objects.

4 Problem Formulation

A physical reasoning task τ is expressed as a set $\{(o_1, o_2, \ldots, o_H), \mathcal{G}, \mathcal{A}\}$ consisting of the goal region, object instances present in the world state and the induced action space. Solving the task implies learning (i) the object/tool types and their locations that provides a construction (an object arrangement for multi-step interaction) and (ii) the control parameters for each object; cumulatively resulting in a high expected reward upon simulating a ball object interaction with the tools. For example, determining the type/sequence of a pendulum, table and a wedge as well as their relative orientation such that when an interacting object like a ball is hit it can fall in a given container (goal region). Formally, our aim is to learn a policy $\pi_\theta(\boldsymbol{a}|\tau)$ that optimizes the following reward objective:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \underset{(\boldsymbol{u},\boldsymbol{v})\sim\pi_\theta(\boldsymbol{a}|\tau)}{\mathbb{E}} \big[\mathcal{R}\big(\text{Simulate}\left((\boldsymbol{u},\boldsymbol{v})|\tau\right)\big)\big] \quad (1)$$

The function $\text{Simulate}(.)$ dynamically simulates the action $\boldsymbol{a}$ sampled from the policy $\pi_\theta(.|\tau)$. First, the simulation *constructs* the object assembly by selecting objects of a particular type, placing them at selected locations, represented by $\mathcal{C}(.)$. Next, the function $f^D(.)$ denotes the *dynamic* simulation of the ball object from the initial state $\mathbf{s}_0$ in which the ball object dynamically interacting with the object assembly. The episode ends when the ball attains a dynamically stable state s^F associated with a reward $\mathcal{R}(s^F)$. The learned policy $\pi_{\theta^*}(\boldsymbol{a}|\tau)$ yields the optimal tool construction and controls $(\boldsymbol{u}, \boldsymbol{v})$ that in effect solves the physical reasoning task.

$$\text{Simulate}(\boldsymbol{a}|\tau) = f^D\big(v_{1:H}, \mathcal{C}(u_{1:H}, \tau)\big). \quad (2)$$

Directly optimizing the policy objective is challenging due to two reasons. First, dynamic simulation $f^D(.)$ is *slow* for dynamic interactions e.g., colliding, sliding etc. are slow causing training a policy via optimization to be very inefficient. Second, the action space a is composed of discrete u and continuous v components, where the continuum of controls for each discrete setting leads to a *large mixed discrete-continuous* search space. Further, the need to model sequential dynamic interactions leads to delayed and a highly sensitive reward making exploration challenging. E.g., changing the relative ordering of tools or slightly perturbing the tool controls (e.g., amplitude of oscillation of a pendulum) can sharply affect the reward at the terminal state of the ball. We address these challenges by first *training physics-informed neural networks for fast simulations of dynamic interactions* instead of relying entirely on high-fidelity simulation. Subsequently, we leverage the learned simulators in *learning a policy that decides tool sequences and controls*.

5 Physics Informed Dynamics Predictors

We train a physics informed neural network (PINN) (Goswami et al. 2020) that predicts the state $\mathbf{s}_t$ of objects undergoing a dynamic interaction at any future time t. The knowledge of governing ordinary differential equations (ODEs) of the form $\dot{\mathbf{s}} = \zeta(\mathbf{s}, \lambda)$, where λ are interaction-specific parameters (e.g., friction coefficient, restitution coefficient), is incorporated via a physics-regularizing term that enforces consistency of predictions with the underlying dynamics. Further, the governing equations are assumed simple and a data term is added that compensates for noise and unmodeled aspects of the dynamic interaction. Finally,

[1]Note: some objects may not be selected in the final assembly, hence may remain in an unused region.

during training, the latent physical parameters are also estimated. Formally, we model PINN $\mathcal{F}_\phi$ with trainable parameters ϕ such that, given the initial environment state $\mathbf{s}_0$, λ, $\mathbf{s}_t$ is approximated as $\mathbf{s}_t \approx \hat{\mathbf{s}}_t := \mathcal{F}_\phi(\mathbf{s}_0, t, \lambda)$. We define $\mathcal{L}_\mathcal{D}(\phi)$ as a data-driven MSE($\mathbf{s}_t$, $\hat{\mathbf{s}}_t$) over data points obtained from a high-fidelity simulator, and $\mathcal{L}_P(\phi, \lambda) =$ MSE($\dot{\hat{\mathbf{s}}}_t, \zeta(\hat{\mathbf{s}}_t, \lambda)$) as the physics-informed loss over collocation points sampled in the input space of the PINN. We obtain $\phi^* = \text{argmin}_\phi [\alpha\mathcal{L}_P(\phi, \lambda) + (1-\alpha)\mathcal{L}_\mathcal{D}(\phi)]$ as the trained parameters of the PINN, with $\alpha \in (0, 1]$ as the weighing factor. We use $\alpha = 0.5$. Additionally, we can estimate unknown λ during training as ϕ^*, $\lambda^* = \text{argmin}_{\phi,\lambda} [\alpha\mathcal{L}_P(\phi; \lambda) + (1-\alpha)\mathcal{L}_\mathcal{D}(\phi)]$. We cascade the PINNs sequentially via approximate, interaction-determining logic to obtain a proxy dynamics simulator for planning, such that

$$\mathbf{s}^F_{\phi^*} = \widehat{\text{Simulate}}(\boldsymbol{a}|\tau) = f^D_{\phi^*}\big(v_{1:H}, \mathcal{C}(u_{1:H}, \tau)\big) \quad (3)$$

where $\mathbf{s}^F_{\phi^*}$ denotes the predicted future state. and $f^D_{\phi^*}$ denotes the proxy dynamic simulation parameterised by ϕ^*.

Trajectory predictions arise from forward pass of a network are rapid but often approximate due to modeling errors. Consequently, the associated *proxy* reward $\mathcal{R}(\mathbf{s}^F_{\phi^*})$ may possess a discrepancy in relation to the actual reward $\mathcal{R}(\mathbf{s}^F)$ at the actual state $\mathbf{s}^F$ predicted by the high-fidelity simulation. This discrepancy is accounted for by modeling the discrepancy, Δr, between the proxy and the actual rewards as a Gaussian Process (Moses et al. 2020), $\Delta r \sim \mathcal{GP}(\boldsymbol{a}|\tau)$. Hence, we define $\tilde{R}(\widehat{\text{Simulate}}(\boldsymbol{a}|\tau)) = \mathcal{R}(\widehat{\text{Simulate}}(\boldsymbol{a}|\tau)) + \Delta r$. As detailed subsequently, we search for tool combinations using the approximate simulations and selective query the real simulator to learn the correction model (see Appendix A for details).

6 Generalized Physical Task Planning

Planning to solve a physical reasoning task involves: (i) planning *which* tool types are to be selected and *where* should they be placed to obtain high-rewards and (ii) deciding controls parameters which determine *how* each tool will interact in sequence. A direct solution search in the joint discrete-continuous space is challenging. Central to the proposed approach is learning a deterministic policy over continuous-space controls via online planning with learned neural predictors. We leverage this procedure in two phases (detailed next) to arrive at a policy that solves a given task. In the first phase, we learn a stochastic policy that yields a representative samples of discrete tool constructions (type and sequence). This is followed by a learning a deterministic policy over continuous controls that refines the coarse predictions by selective queries to a high-fidelity simulator.

6.1 Soft-adaptive MCTS for Tool Controls

This section focuses the estimation of control variables for a given tool sequence. Formally, we seek to estimate a (deterministic) policy that optimizes the following objective:

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \underset{\boldsymbol{v}\sim\pi_\theta(v|u,\tau)}{\mathbb{E}} \big[\tilde{\mathcal{R}}\big(\widehat{\text{Simulate}}\,(\boldsymbol{v}|\boldsymbol{u}, \tau)\big)\big]. \quad (4)$$

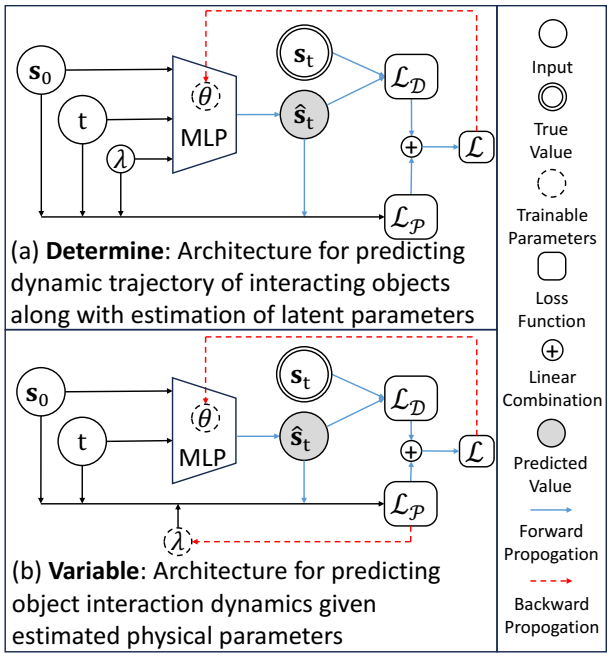


Figure 3: Architecture of the Physics-Informed Dynamics Predictors under two different scenarios: (a) The interaction-specific parameters (λ) are unknown and are learnt while training the network on collected data. The physics-informed loss is based on an estimate of λ therefore, backpropagation updates λ until convergence. (b) The interaction-specific parameters (λ) are known, and the network is trained with λ as input, enabling generalization across a range of λ values. After estimation of λ by network in (a), we train network in (b) and use the latter for trajectory prediction.

We adopt a model-based approach to learn this policy via forward search. A Monte Carlo Tree Search (MCTS) procedure is used to plan the control variables $\{v_0, v_1, \ldots, v_H\}$ by iteratively simulating multi-step interactions using the learned neural predictors. For example, for a tool sequence comprising of the pendulum, table and wedge, the learned predictors are cascaded to predict the successive ball interactions from one contact till another until the ball is dynamically stable. During search Upper Confidence Bound criteria (Kocsis and Szepesvári 2006) is applied to the look-ahead tree for state selection. Since simulations extend till a finite horizon of H steps, no further roll outs are required and the terminal reward is back-propagated (see Algorithm 1).

The standard MCTS is augmented in two ways to improve convergence and solution search for our problem domain. First, a softmax weighting is applied to the rewards while estimating the value of a state-action pair. Since the rewards are sensitive where minor perturbations in controls lead to large reward changes, the softmax weighting was empirically found effective in rapidly guiding the search towards high rewards. Second, the search process adapts the inaccurate estimates of the final states (leading to incorrect reward estimates) arising from neural simulations. The incorrect re-

Algorithm 1: Soft-adaptive Monte Carlo Tree Search

Require: Task τ, Tool construction $\boldsymbol{u}$, $\widehat{\text{Simulate}}(\cdot|\tau)$, $\mathcal{GP}$, adapt flag adapt
{adapt = False while training θ^*, else True}

Initialize search tree with empty root; $\boldsymbol{v}^*$
for control index $m = 1$ to H **do**
  $\mathcal{C}_m \leftarrow \{\boldsymbol{v}_m^{(0)}, \ldots, \boldsymbol{v}_m^{(\mathcal{D})}\}$ are the candidate controls
  Initialize $\mathcal{R}_i \leftarrow \{\,\}\ \forall \boldsymbol{v}_m^{(i)} \in \mathcal{C}_m$
  for iteration $k = 1$ to $\mathcal{K}$ **do**
    $\forall \boldsymbol{v}_m^{(i)} \in \mathcal{C}_m$: $\text{UCB}(\boldsymbol{v}_m^{(i)}) \leftarrow \hat{V}_i + c\sqrt{\frac{\log N}{n_i+\epsilon}}$
    $\boldsymbol{v}_m^* \leftarrow \text{argmax}_{\boldsymbol{v}_m^{(i)} \in \mathcal{C}_m} \text{UCB}(\boldsymbol{v}_m^{(i)})$
    Sample controls $\tilde{\boldsymbol{v}}_{m+1:H}$ uniformly (within range)
    $\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v}_{1:m-1}^*, \boldsymbol{v}_m^*, \tilde{\boldsymbol{v}}_{m+1:H}$
    Compute Proxy reward: $\hat{r} \leftarrow \mathcal{R}(\widehat{\text{Simulate}}((\boldsymbol{u}, \hat{\boldsymbol{v}})|\tau))$
    if adapt **then**
      $\mu, \sigma \leftarrow \mathcal{GP}((\boldsymbol{u}, \hat{\boldsymbol{v}})|\tau)$ { *// $\mathcal{GP}-UCB$ Criteria*}
      $\hat{r} \leftarrow \hat{r} + \mu + \beta^{1/2}\sigma$
    end if
    $\mathcal{R}_{\boldsymbol{v}_m^*} \leftarrow \mathcal{R}_{\boldsymbol{v}_m^*} \cup \{\hat{r}\}$
    $N \leftarrow N + 1$; $n_{\boldsymbol{v}_m^*} \leftarrow n_{\boldsymbol{v}_m^*} + 1$
    $\hat{V}_{\boldsymbol{v}_m^*} \leftarrow \frac{\sum_j e^{\mathcal{R}_{\boldsymbol{v}_m^*}[j]} \cdot \mathcal{R}_{\boldsymbol{v}_m^*}[j]}{\sum_j e^{\mathcal{R}_{\boldsymbol{v}_m^*}[j]}}$ { *// Softmax Weighting*}
  end for
  $\boldsymbol{v}_{1:m}^* \leftarrow \boldsymbol{v}_{1:m-1}^*, \text{argmax}_{\boldsymbol{v}_m^{(i)} \in \mathcal{C}_m} \text{UCB}(\boldsymbol{v}_m^{(i)})$
end for
return $\boldsymbol{v}_{1:H}^*$

wards are adapted by querying a $\mathcal{GP}$ as in (Srinivas et al. 2010) to provide a correction as well as update the $\mathcal{GP}$ using selective roll outs in the high-fidelity simulator.

6.2 Generative Flows Networks for Tool Selection

With the ability to determine control variables for a given tool construction, we turn our attention to reasoning over the sequence of objects and their types. The problem can be cast as generating a construction over discrete variables $\{u_1, \ldots u_H\}$. Since the combinatorial space of possible tool sequences is large, we seek to sample a *candidate* set of high-reward construction. We cast this task as learning a stochastic policy for generating a tool sequence proportional to the reward. The reward from $\widetilde{\text{Simulate}}$ is obtained by creating the assembly using $\mathcal{C}(u_0, u_1, \ldots, u_H)$. This is followed by performing MCTS exploration for controls (as outlined in the previous section) using neural dynamics simulations. Note the simulations are without the Gaussian Process reward correction from high-fidelity simulation for improving learning efficiency. The reward associated with the predicted state is termed as "proxy" reward (fast but inaccurate w.r.t. high-fidelity simulation).

$$\theta^* = \underset{\theta}{\text{argmax}} \underset{\boldsymbol{u} \sim \pi_\theta(u|\tau)}{\mathbb{E}} \left[\mathcal{R}\left(\widetilde{\text{Simulate}}\left(\boldsymbol{u}|\tau\right)\right)\right] \tag{5}$$

Policy learning is cast as a Generative Flow Network (GFlowNet) (Bengio et al. 2021) that results in a representative set of constructions with high values of the proxy reward, in effect, allowing rapid exploration around the modes of the reward distribution. The policy is trained by optimizing a flow preserving objective that enforces a TD-style relationship (Bengio et al. 2021).

$$\pi_{\theta^*}\left(\boldsymbol{u}|\tau\right) = \frac{\mathcal{R}(\widetilde{\text{Simulate}}(\boldsymbol{u}|\tau))}{\sum_{\bar{\boldsymbol{u}}} \mathcal{R}(\widetilde{\text{Simulate}}(\bar{\boldsymbol{u}}|\tau))} \tag{6}$$

The learned stochastic policy $\pi_{\hat{\theta}}\left(\boldsymbol{a}|\tau\right)$ expresses a distribution over the tool construction (sampled proportional to the estimated proxy reward). Examining the top-k samples results in a sequence $\{(\hat{\boldsymbol{u}}_1, \hat{\boldsymbol{v}}_1), \ldots, (\hat{\boldsymbol{u}}_K, \hat{\boldsymbol{v}}_K)\}$ which is projected to the tool construction set $\hat{\mathbf{U}} = \{\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_K\}$ as the set of k representative hypothesis for tool construction. The diverse set of samples obtained are refined with a procedure that selectively queries the real simulator to improve its reward computation (outlines in the next sub-section). Since the neural simulations are approximate, the batch of samples allow the successive solution guide the search towards likely candidates. Note that the most likely solution obtained from GFlowNets (using the proxy reward) may not be the best reward candidate once the actual rewards are obtained in the subsequent step. Hence, having a diverse set of samples from an exploration policy is beneficial. Additional details for the PhyPlan model appear in Appendix A.

7 Evaluation Methodology

Simulation Environment and Tasks. We lift physical reasoning tasks from prior 2D environments (Allen, Smith, and Tenenbaum 2020; Bakhtin et al. 2019) into a challenging 3D setting with a simulated manipulator, increasing the complexity of planning, control, and perception. The five tasks: *Launch*, *Slide*, *Bounce*, *Bridge*, and *Ricochet*, evaluate the long-horizon reasoning and are implemented in Isaac Gym (NVIDIA 2022) using the *Franka Emika* robot manipulators interacting with objects within range. The objective is to guide a ball to a randomly placed box that cannot be directly moved. The spatial gap and constraints require multi-step dynamic interactions and tool-mediated reasoning. The 3D setting introduces additional challenges, including occlusion, depth ambiguity, and actuation noise. Perception is enabled via a simulated RGB camera; objects are identified using frozen visual-language models (Kirillov et al. 2023; Liu et al. 2023), and poses are recovered using known intrinsics. This setup grounds the physical reasoning in realistic 3D interactions, without relying on abstract dynamics. Also see Figure 2 in Appendix B for additional details.

Evaluation Metric and Baselines. We compare PhyPlan against several baselines to evaluate its efficiency, generalizability, and performance, measured in terms of a regret measure, $e = (r^* - r)/r^*$ (Moses et al. 2020), where r^* denotes the maximum attainable reward ($r^* = 1$ for the aforementioned tasks) and r the achieved reward. The baselines selected are as follows: (a) **Random**: samples actions uniformly at random; (b) **LLM**: uses a Large Language Model

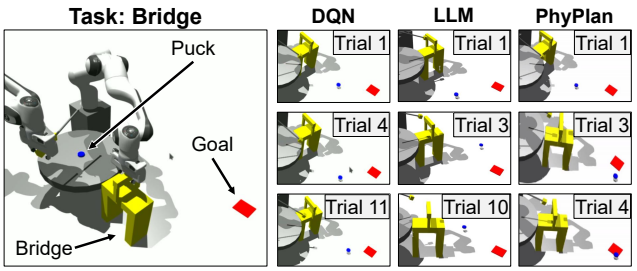


Figure 4: Qualitative Plan Comparison: The planning trials performed by DQN, LLM, and PhyPlan are compared in *Bridge* task. DQN does not use a bridge even after 11 trials, while LLM does not align it correctly. PhyPlan discovers its use in the second trial and learns to aligns it in the fourth trial to achieve the goal.

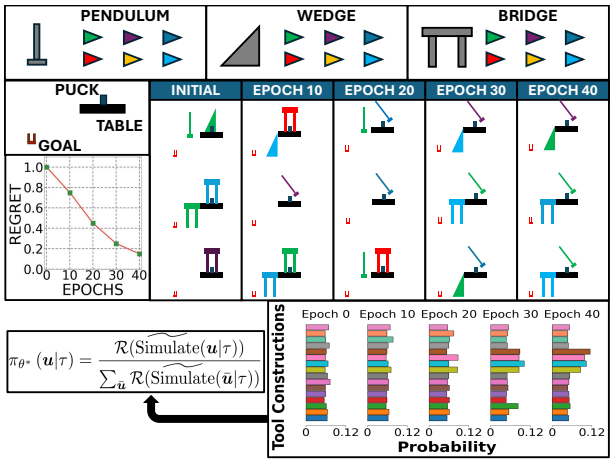


Figure 5: Progressively reasoning over tools: PhyPlan's tool selector based on GFlowNet eventually learns to select and place the appropriate tool classes and the best corresponding tool variants.

(Gemini 2.5 Pro (Google 2023)) to generate actions iteratively based on task descriptions and reward feedback; (c) **DQN**: a Deep Q-Network trained on 1000 random state-action-reward samples, with policy inferred by sampling 1000 actions and selecting the best (Bakhtin et al. 2019); (d) **PPO**: a Proximal Policy Optimization agent (Schulman et al. 2017) trained similarly to DQN and evaluated via best-of-1000 sampling; and (e) **MBPO**: a model-based policy optimizer inspired by PILCO (Deisenroth and Rasmussen 2011), where a $\mathcal{GP}$ predicts action values and gradients are used to optimize a policy. These baselines are evaluated in a control-selection setting with access to "ideal" tool and referred to as *control-selector* baselines. Note that all approaches except LLM and Random adapt via $\mathcal{GP}$. LLM is given information to adapt via successive prompting as described in appendix C. PhyPlan, in contrast, *jointly* reasons over both discrete tool selection and continuous control, enabling hybrid decision-making in complex physical tasks. We also compare against an *MCTS-based Tool+Control Selector*, the only other baseline capable of joint reasoning, where the GFlowNet-based tool selector in PhyPlan is replaced with an MCTS-based alternative. Additional details regarding the baselines are in Appendix C. Hyperparameters are estimated via grid search. For Soft-adaptive MCTS, parameters are $D = 20$, $\mathcal{K} = 25$, $c = 0.7$. PINN hyperparameters and training configurations (Table 2, Table 3), and other implementation details appear in appendix.

8 Experiments & Results

We evaluate the planning proficiency of PhyPlan in multi-interaction tasks requiring physical reasoning over a hybrid space of discrete tool choices and continuous control parameters. Table 1 reports regret across five planning trials. Notably, all baselines, except PhyPlan and the MCTS-based Tool+Control Selector, are provided with *ideal tools* and tasked with only selecting control parameters. Despite this advantage, our approach outperforms all baselines by some margin. LLMs perform worse than random, likely due to limited ability to model physical dynamics. RL methods like PPO (Schulman et al. 2017) and DQN, though better than LLMs, still produce poor results. In contrast, PhyPlan jointly reasons over both tool selection and control, achieving the *lowest regret* across all tasks. This demonstrates the strength of structured reasoning, with GFlowNet guiding exploration in the discrete tool space and MCTS refining continuous control decisions. While the MCTS-based Tool+Control Selector also addresses hybrid reasoning, its performance is significantly lower than PhyPlan, highlighting the role of guided exploration and reward alignment.

Figure 4 shows a qualitative comparison on the challenging Bridge task. PhyPlan solves the task in fewer trials and demonstrates long-horizon reasoning by learning to place and use the bridge effectively, a known challenge for model-free methods (Allen, Smith, and Tenenbaum 2020). Figure 5 visualizes PhyPlan's training dynamics on the Bridge task, showing top-3 tool selections every 10 epochs. The model first discovers optimal tool ordering and later refines control strategies, eventually reasoning to place a pendulum on a table followed by a bridge, resulting in successful task completion. How the model adjusts the continuous control variables in qualitatively shown in Fig. 6 and in Appendix D. Figures 7, 8 further illustrate the execution on other tasks.

8.1 Comparison of Learning Times

We compare policy learning times across all methods in Figure 9, which plots the mean regret achieved versus reasoning time, averaged over our physical reasoning tasks. PhyPlan achieves a *45% lower regret* than the best-performing baseline, DQN. We reiterate that DQN, PPO, and MBPO are control-selector baselines as they are provided with ideal tools and only optimize over continuous control parameters. In contrast, PhyPlan jointly reasons over discrete tool selection and continuous controls within a hybrid decision space. Despite this increased complexity, PhyPlan has comparable runtime to DQN and PPO, while consistently outperforming them across all tasks. Compared to MBPO, a strong model-based method designed for high-dimensional contin-

Approach	Average Regret (↓) ± SE (↓)				
	Bounce Task	**Launch Task**	**Slide Task**	**Bridge Task**	**Ricochet Task**
PhyPlan[1]	**0.08 ± 0.01**	**0.05 ± 0.01**	**0.29 ± 0.03**	**0.15 ± 0.06**	**0.56 ± 0.10**
MCTS-based Tool+Control Selector[1]	0.20 ± 0.01	0.17 ± 0.01	0.49 ± 0.05	0.42 ± 0.06	0.73 ± 0.09
Random[2]	0.42 ± 0.04	0.40 ± 0.08	0.46 ± 0.07	0.27 ± 0.03	1.53 ± 0.15
LLM[2]	0.41 ± 0.07	0.40 ± 0.08	0.47 ± 0.08	0.31 ± 0.06	1.90 ± 0.22
PPO[2]	0.29 ± 0.05	0.35 ± 0.05	0.45 ± 0.08	0.44 ± 0.03	1.11 ± 0.11
DQN[2]	**0.08 ± 0.02**	0.36 ± 0.05	0.56 ± 0.10	0.25 ± 0.06	0.83 ± 0.07
MBPO[2]	0.48 ± 0.07	0.52 ± 0.10	0.58 ± 0.07	0.63 ± 0.09	1.72 ± 0.17

Table 1: A comparison of average regret values for benchmark tasks PhyPlan and other baseline approaches. ↓ signifies that lower avg. regret and SE are better. [1]PhyPlan and MCTS-based Tool+Control Selector are tool+control selector baselines, i.e., they reason jointly over tools and their controls. [2]Control selector baselines are given "ideal" tools and select for their controls.

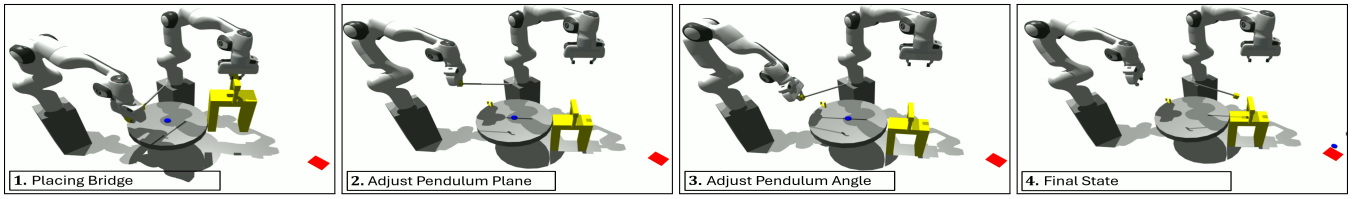


Figure 6: Execution of the Policy learned for the Bridge Task: After 4 trials, the robot learns to place the bridge in the path of the box and releases the pendulum from the correct location to slide the box to the goal.

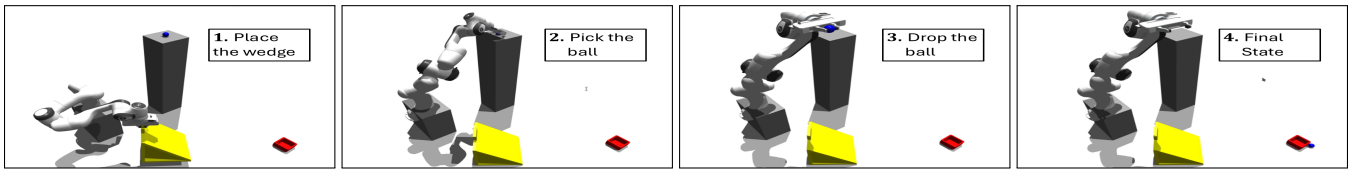


Figure 7: Execution of the Policy learned for the Bounce Task: After 3 trials, the robot learns a policy that correctly aligns the wedge's orientation and drops the ball from a precise height over the wedge to land it into the box.

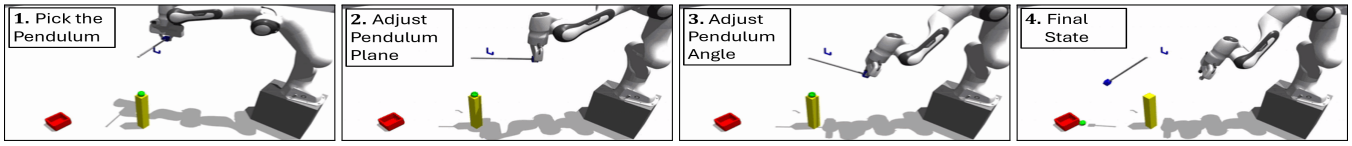


Figure 8: Execution of the Policy learned for the Launch Task: After 2 trials, the robot learns a policy that correctly aligns the pendulum's plane and angle to throw the ball into the box.

uous spaces, PhyPlan is not only 3× *faster*, but also achieves a *70% improvement in regret measure*. These results underscore the efficiency and effectiveness of the proposed approach in solving physical reasoning tasks.

8.2 Ablation Study

The proposed PhyPlan comprises of physics-informed dynamics predictor, $\mathcal{GP}$-based adaptation to new real environment, and a novel soft-adaptive MCTS, all of which play a critical role in its overall performance. Therefore, we evaluate the performance of the individual components to better understand their contributions to the overall performance.

Physics-informed Dynamics Predictor. The physics-informed dynamics predictor in PhyPlan serves two key purposes: (a) enabling fast trajectory prediction to efficiently evaluate actions, and (b) estimating non-visual physical parameters (e.g., coefficient of friction) that are not directly observable. Figure 10(a) shows that this predictor consistently outperforms purely data-driven models. Importantly, it is also significantly *faster* than the fine-grained simulator, contributing directly to PhyPlan's overall efficiency as reflected in Figure 9. Figure 10(b) demonstrates its accuracy in inferring the coefficient of friction in sliding scenarios. Also see Appendix D.

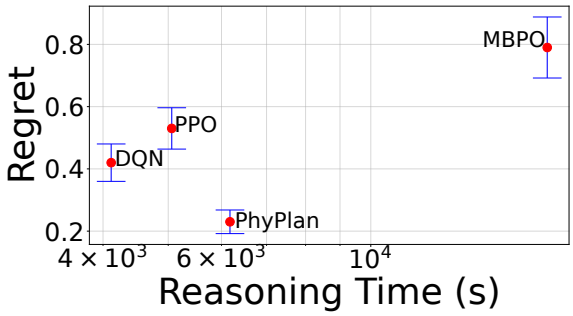
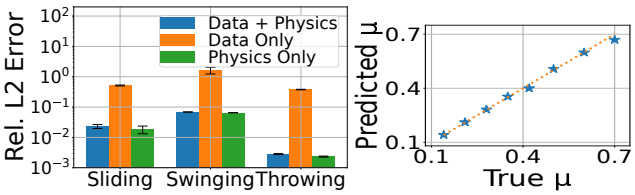


Figure 9: Average Regret vs. Reasoning Time: PhyPlan, reasoning over tool construction *and* controls, accomplishes lower regret than approaches that only select controls.



(a) Trajectory prediction (b) Parameter estimation

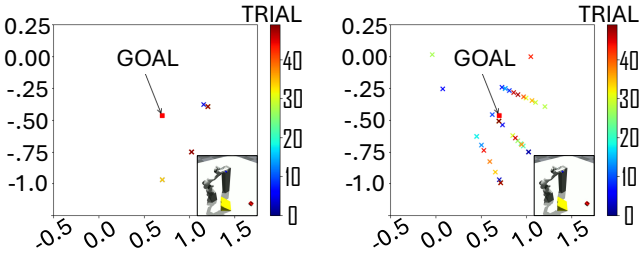
Figure 10: Accuracy of Dynamics Predictors in (a) trajectory prediction (extrapolation) and (b) latent parameter estimation for sliding task. For trajectory estimation, physics-based approach outperforms purely data-driven counterpart.

$\mathcal{GP}$-based Reward Corrections. Figure 11 compares PhyPlan with its ablated variant, PhyPlan-No Adapt, which excludes the $\mathcal{GP}$-based online adaptation module. The figure depicts the final ball positions after each trial in the *Bounce* task. Without adaptation, PhyPlan-No Adapt fails to explore effectively due to its lack of awareness of real-world discrepancies. In contrast, PhyPlan leverages prediction error to quickly refine its policy and converge toward the goal. The $\mathcal{GP}$ component learns localized reward discrepancies around high-performing actions, enabling targeted updates to the reward model and accelerating convergence.

Soft-adaptive MCTS. We illustrate the performance of soft-adaptive MCTS as opposed to the conventional MCTS for the Ricochet Task in Figure 12. We observe that the proposed soft-adaptive variant yield lower regret and faster convergence (fewer iterations), also see Appendix D.

9 Conclusions & Future Directions

We considered the problem of training a robot to perform goal-directed reasoning on unseen tasks requiring a composition of physical skills. We efficiently learn dynamics predictors leveraging the governing equations and embed them into an MCTS procedure that utilizes it to perform rapid rollouts during reasoning (as opposed to a slow full-scale physics simulation engine). GFlowNets are employed to sample the best tool instantiation, which uses the described MCTS as a proxy reward model. The MCTS procedure is further used to plan the controls of the selected



(a) PhyPlan - No Adapt (b) PhyPlan

Figure 11: Effect of $\mathcal{GP}$-based adaptation of reward model: Figure shows the ball's coordinates relative to the goal in *Bounce* task for 50 trials. The adaptation step allows compensation for modeling errors in the dynamics models when used sequentially.

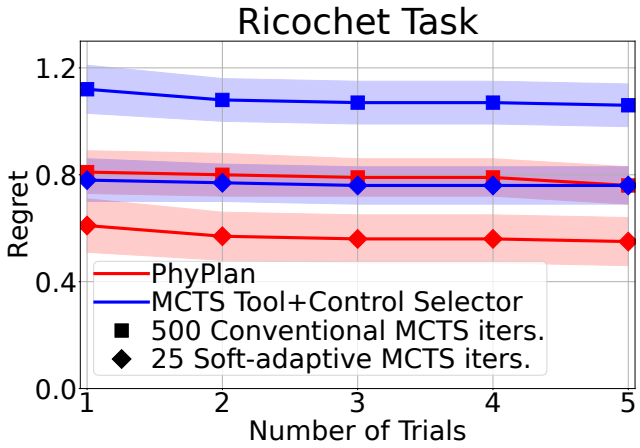


Figure 12: Soft-adaptive vs. Conventional MCTS: Switching to a soft-maximum value estimation in MCTS yields a significant improvement in efficiency as well as regret scores in our hardest physical reasoning task: *Ricochet* Task.

tools with online adaptations via $\mathcal{GP}$ to compensate for the modeling errors through selective real rollouts.

PhyPlan is data efficient and improves over *physics-uninformed* approaches. However, it has certain limitations: (i) the curriculum of foundational skills in tasks is assumed and (ii) the physics of the skills is known, not discovered; addressing these remains part of future work. Moreover, while Isaac Gym simulations incorporate realistic robotic constraints, robot experiments and validation on real data remains part of future work.

Acknowledgements

We thank Yardi School of Artificial Intelligence (ScAI) for research grant under Wipro COE on Generative AI and the conference participation support. We acknowledge support from the ICMR National Center for Assistive Health Technologies (NCAHT), Government of India. Authors thank the IIT Delhi High Performance Computing (HPC) facility, the Dept. of Computer Science and Engineering (CSE) and the School of Information Technology (SIT) for computing sup-

port. We thank Mr. Anil Sharma, Ms. Sunita Negi and Mr. Shailendra Negi for technical/administrative support.

References

Allen, K. R.; Lopez-Guevara, T.; Stachenfeld, K.; Sanchez-Gonzalez, A.; Battaglia, P.; Hamrick, J.; and Pfaff, T. 2022. Physical design using differentiable learned simulators. *arXiv preprint arXiv:2202.00728*.

Allen, K. R.; Smith, K. A.; and Tenenbaum, J. B. 2020. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47): 29302–29310.

Bakhtin, A.; van der Maaten, L.; Johnson, J.; Gustafson, L.; and Girshick, R. 2019. Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, 32.

Bear, D. M.; Wang, E.; Mrowca, D.; Binder, F. J.; Tung, H.-Y. F.; Pramod, R.; Holdaway, C.; Tao, S.; Smith, K.; Sun, F.-Y.; et al. 2021. Physion: Evaluating physical prediction from vision in humans and machines. *Advances in Neural Information Processing Systems*.

Bengio, E.; Jain, M.; Korablyov, M.; Precup, D.; and Bengio, Y. 2021. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. arXiv:2106.04399.

Bengio, Y.; Lahlou, S.; Deleu, T.; Hu, E. J.; Tiwari, M.; and Bengio, E. 2023. GFlowNet Foundations. *Journal of Machine Learning Research*, 24(210): 1–55.

Chakraborty, S. 2020. Simulation free reliability analysis: A physics-informed deep learning based approach. *arXiv preprint arXiv:2005.01302*.

Chen, Y.; Wu, T.; Wang, S.; Feng, X.; Jiang, J.; Lu, Z.; McAleer, S.; Dong, H.; Zhu, S.-C.; and Yang, Y. 2022. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 5150–5163.

Davis, E. 2008. Chapter 14 Physical Reasoning. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, 597–620. Elsevier.

Deisenroth, M. P.; and Rasmussen, C. E. 2011. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 465–472.

Finn, C.; and Levine, S. 2017. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2786–2793. IEEE.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of Artificial Intelligence Research*, 27: 235–297.

Gao, J.; Sarkar, B.; Xia, F.; Xiao, T.; Wu, J.; Ichter, B.; Majumdar, A.; and Sadigh, D. 2024. Physically Grounded Vision-Language Models for Robotic Manipulation. In *International Conference on Robotics and Automation (ICRA)*.

Google. 2023. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805.

Goswami, S.; Anitescu, C.; Chakraborty, S.; and Rabczuk, T. 2020. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106: 102447.

Hamrick, J. B.; Allen, K. R.; Bapst, V.; Zhu, T.; McKee, K. R.; Tenenbaum, J. B.; and Battaglia, P. W. 2018. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023. Reasoning with Language Model is Planning with World Model. arXiv:2305.14992.

Ijspeert, A. J.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; and Schaal, S. 2013. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–373.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643*.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40: e253.

Liu, S.; Zeng, Z.; Ren, T.; Li, F.; Zhang, H.; Yang, J.; Li, C.; Yang, J.; Su, H.; Zhu, J.; et al. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*.

Moses, C.; Noseworthy, M.; Kaelbling, L. P.; Lozano-Pérez, T.; and Roy, N. 2020. Visual Prediction of Priors for Articulated Object Interaction. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 10480–10486. IEEE.

NVIDIA. 2022. Isaac Gym: High-Performance GPU-Based Physics Simulation For Robot Learning.

Park, D.; Noseworthy, M.; Paul, R.; Roy, S.; and Roy, N. 2020. Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning. In *Conference on robot learning*, 1005–1014. PMLR.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Schwettmann, S.; Fischer, J.; Tenenbaum, J.; and Kanwisher, N. 2018. Evidence for an Intuitive Physics Engine in the Human Brain. In *CogSci*.

Srinivas, N.; Krause, A.; Kakade, S.; and Seeger, M. 2010. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th In-*

ternational Conference on International Conference on Machine Learning, ICML'10, 1015–1022. Madison, WI, USA: Omnipress. ISBN 9781605589077.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Toussaint, M.; Allen, K. R.; Smith, K. A.; and Tenenbaum, J. B. 2019. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning - Extended Abtract. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 6231–6235. International Joint Conferences on Artificial Intelligence Organization.

Wu, J.; Yildirim, I.; Lim, J. J.; Freeman, B.; and Tenenbaum, J. 2015. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in neural information processing systems*, 28.

Xue, C.; Pinto, V.; Gamage, C.; Nikonova, E.; Zhang, P.; and Renz, J. 2023. Phy-Q as a measure for physical reasoning intelligence. *Nature Machine Intelligence*, 5(1): 83–93.

Zeng, A.; Song, S.; Lee, J.; Rodriguez, A.; and Funkhouser, T. 2020. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319.