M4PO: Massively Multi-Task Multi-Embodiment Model-Based Policy Optimization

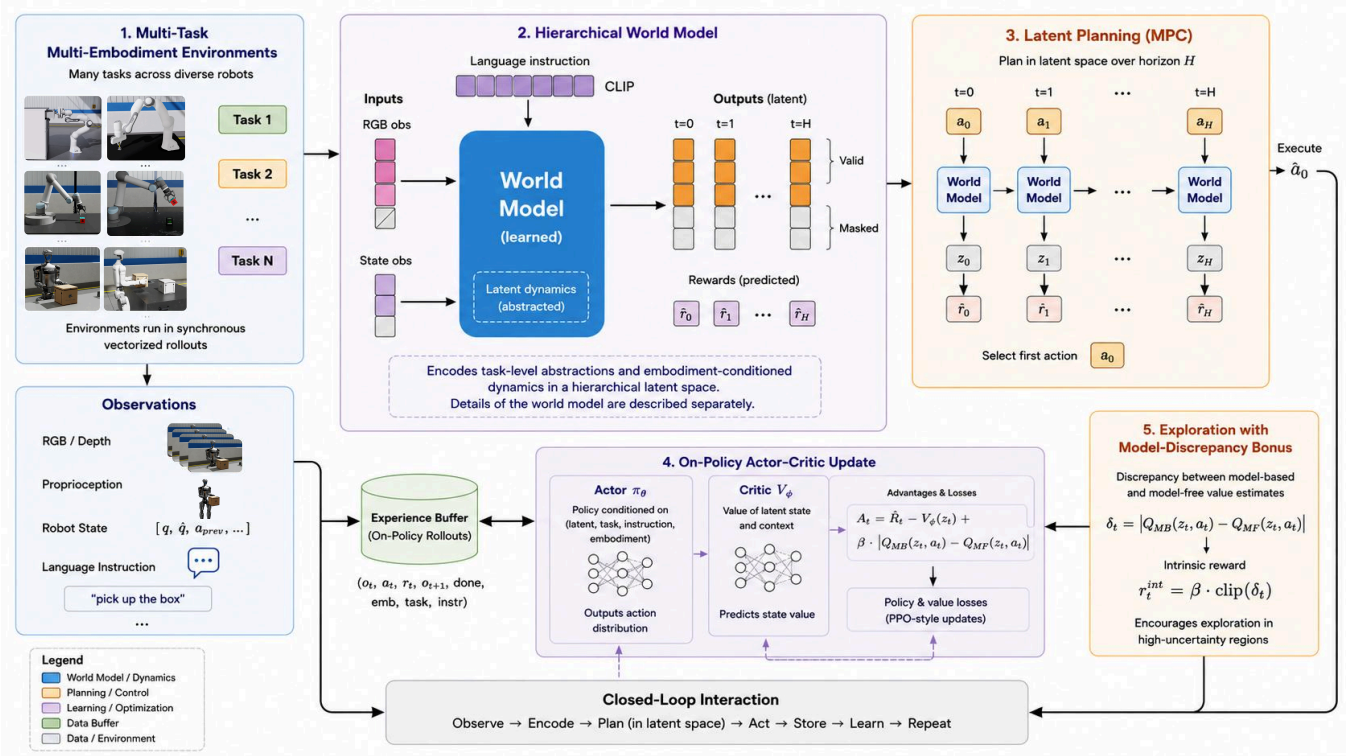


Fig. 1. Overview of M4PO: shared multimodal world modeling, latent MPPI planning, and on-policy optimization. RGB-D, structured state, proprioception, task language, and embodiment context condition the model and policy. Sec. III specifies the complete objective and planner likelihood.

***Abstract*— We present M4PO, a multi-task multi-embodiment manipulation framework combining hierarchical world modeling, stochastic latent-space planning, and on-policy reinforcement learning. RGB-D, structured state, proprioception, and CLIP-encoded language condition task- and embodiment-dependent latent dynamics. Embodiment-masked continuous action tokenization normalizes and encodes control coordinates, masking invalid dimensions during prediction, planning, and optimization. A model-based/model-free value-discrepancy bonus augments rewards; fresh rollouts update the world model and planner-policy. On the main multi-embodiment benchmark, M4PO achieves 54.3% average and 45.1% worst-embodiment success, exceeding NEWT by 15.2 and 12.7 percentage points. Unitree G1 and Franka hardware evaluation achieves 85 successes in 100 trials across lifting, fruit manipulation, and stacking, without real-world policy updates.**

I. Introduction

Generalist manipulation requires knowledge sharing across tasks and morphologies while respecting kinematics, actuator limits, and contact dynamics. Partial observability [1] and physical deployment [2] add further challenges. A shared policy must preserve embodiment-specific information to generate feasible actions.

M4PO learns a shared world model and planner-policy from parallel simulation. RGB-D, structured state, proprioception, and CLIP-encoded instructions form its multimodal interface. Task- and embodiment-conditioned latent components are intended to specialize in task progression and morphology-specific dynamics while exchanging information through prediction. Embodiment-masked continuous action tokenization normalizes and encodes controls to $D_a = \max_e d_a^e$; prediction, planning, likelihoods, and execution respect valid coordinates and each robot's control semantics.

Building on M3PO [3], M4PO optimizes the stochastic MPPI distribution used for rollout collection. Value discrepancy encourages exploration, and fresh rollouts update both policy and world model.

We evaluate fixed-base, dexterous, and humanoid manipulation, including expanded IsaacLab-Arena experiments. The main shared-policy benchmark improves average success over NEWT by 15.2 percentage points. Hardware tests cover G1 box lifting and Franka fruit picking, sorting, and stacking. G1 uses nonvisual inputs after simulation adaptation; hardware policy parameters remain fixed. Fruit sorting tests

a composition absent from simulation training.

II. Related Work

Dreamer methods [4]–[6] optimize behavior through imagined latent trajectories. TD-MPC [7] and TD-MPC2 [8] combine temporal-difference learning with latent predictive control; NEWT [9] extends shared world modeling to language-conditioned multi-task control with demonstration pretraining and online interaction. I-JEPA [10] learns predictive abstract representations. M3PO [3] links model-based and model-free learning through value discrepancy. M4PO adds hierarchical task–embodiment dynamics and masked controls for shared robot learning.

OpenVLA [11], GR00T [12], and the π family [13] use pretrained representations and broad robot datasets for generalist manipulation. These methods motivate language conditioning and cross-embodiment sharing; M4PO studies improvement through simulation interaction and learned-model planning.

Action chunking [14] models control sequences for coordinated manipulation, while diffusion policies [15] represent multimodal visuomotor actions. Their demonstration coverage shapes performance. M4PO instead learns from reinforcement-learning interaction, updating its dynamics model and planner-policy before hardware deployment.

M3PO [3] provides the on-policy planning and discrepancy-learning foundation; M4PO adds explicit task–embodiment transition structure. NEWT [9] combines demonstration pretraining with replay-based learning, while M4PO updates its world model and planner-policy on fresh rollouts. Our contribution centers on coupling embodiment-to-task prediction with an embodiment-masked continuous action interface and likelihood-based optimization of the executed planner distribution.

III. M4PO: Method

Figure 1 summarizes M4PO. Extending M3PO [3], the hierarchy conditions task progression on predicted embodiment dynamics, coupling g_t and b_t in one shared model. State padding and embodiment-masked continuous action tokenization provide a shared interface. The planner defines the executed squashed-Gaussian policy over valid bounded actions, whose likelihood is optimized directly. Both model and policy use fresh rollout batches, differing from NEWT's demonstration-pretrained, replay-based learning [9].

A. Multimodal Encoding and Hierarchical Dynamics

Fig. 2 shows the shared encoder, hierarchical dynamics, and prediction heads.

1) Observations and context: For task $\tau \in \mathcal{T}$ and embodiment $e \in \mathcal{E}$, $o_t^e = (I_t^e, p_t^e, s_t^e)$ contains RGB-D observations, proprioception, and structured robot/task state. The task instruction ℓ_τ and embodiment identity provide contexts $c_\tau = E_{\text{lang}}(\ell_\tau)$ and $c_e = E_{\text{emb}}(e)$. We use a frozen CLIP text encoder; language ablations replace task text with learned task-ID embeddings or remove it.

2) Embodiment-masked continuous action tokenization: Embodiment e has d_a^e valid control coordinates. The continuous action tokenizer maps these coordinates to the shared dimension $D_a = \max_e d_a^e$. The policy samples $v_t^e \in \mathbb{R}^{d_a^e}$ and executes

$$u_t^e = \tanh(v_t^e), \qquad \bar{a}_t^e = P_a^e(u_t^e) \odot m_a^e,$$
$$a_t^e = (\mathcal{N}_a^e)^{-1}(U_a^e(\bar{a}_t^e)).$$

Here $\mathcal{N}_a^e$ normalizes physical controls to $[-1,1]^{d_a^e}$, P_a^e tokenizes to D_a, m_a^e masks invalid tokens, and U_a^e extracts valid coordinates. Invalid coordinates are zeroed before prediction and execution. Proprioception and structured state are normalized, padded, and masked analogously; m_o^e collects their validity masks.

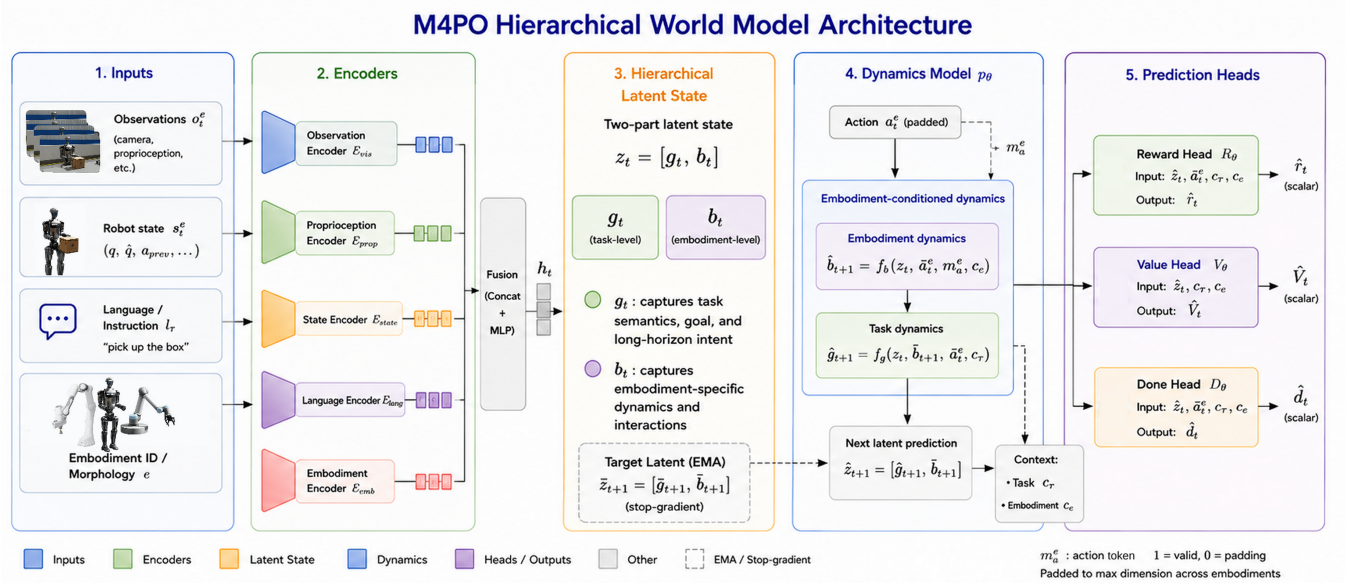


Fig. 2. Hierarchical world-model architecture. Separate observation and context encoders feed task- and embodiment-conditioned latent components. The predicted embodiment transition conditions the task-component prediction; separate heads predict reward, extrinsic value, and termination.

3) Control-rate alignment: A fixed interpolation layer maps each policy decision to K_e low-level controller updates. The corresponding reward and termination flag are

$$r_t^\tau = \sum_{q=0}^{K_e-1} (\gamma_{\text{low}}^e)^q r_{t,q}^{\text{low}},$$
$$d_t = \max_{0 \le q < K_e} d_{t,q}^{\text{low}}.$$

When low-level discounting is used, choose $(\gamma_{\text{low}}^e)^{K_e} = \gamma$; otherwise set $\gamma_{\text{low}}^e = 1$ and discount between decisions. Training transitions are $(o_t^e, \bar{a}_t^e, r_t^\tau, d_t, o_{t+1}^e, \tau, e)$.

4) Hierarchical world model: Encoded visual, proprioceptive, and structured-state features are concatenated as u_t^{obs}. An MLP fuses these features with both contexts and forms the latent state:

$$h_t = F_\theta([u_t^{\text{obs}}, c_\tau, c_e]),$$
$$g_t = \phi_g(h_t, c_\tau), \qquad b_t = \phi_b(h_t, c_e),$$
$$z_t = [g_t, b_t] = \phi_\theta(o_t^e, c_\tau, c_e).$$

Components g_t and b_t are intended to capture task progression and morphology-specific dynamics, respectively. Joint encoding shares information; prediction proceeds from embodiment to task:

$$\hat{b}_{t+1} = f_b(z_t, \bar{a}_t^e, m_a^e, c_e),$$
$$\hat{g}_{t+1} = f_g(z_t, \hat{b}_{t+1}, \bar{a}_t^e, c_\tau).$$

Together, these define $\hat{z}_{t+1} = [\hat{g}_{t+1}, \hat{b}_{t+1}] = f_\theta(z_t, \bar{a}_t)$. Below, fixed contexts, masks, and embodiment superscripts are implicit, and r_t denotes r_t^τ.

Reward, extrinsic-value, and termination heads predict $\hat{r}_t = R_\theta(z_t, \bar{a}_t)$, $\hat{V}_t^{\text{ext}} = V_\theta^{\text{ext}}(z_t)$, and $\hat{d}_t = D_\theta(z_t, \bar{a}_t)$, respectively. The termination head uses a sigmoid output. An EMA target encoder supplies $\bar{z}_t = \phi_{\bar{\theta}}(o_t)$ for the masked training objectives defined next.

B. World-Model Targets and Terminal-Masked Losses

The world model is updated on the freshly collected on-policy batch $\mathcal{B}_k$. The EMA parameters $\bar{\theta}$ include the target encoder and extrinsic value head. For a length-H segment, write $\bar{V}_s = V_{\bar{\theta}}^{\text{ext}}(\bar{z}_s)$ and $y_s = y_s^{\lambda_{\text{wm}}}$. The finite-horizon value target is

$$y_{t+H} = \bar{V}_{t+H},$$
$$y_s = r_s + \gamma(1-d_s)\big[(1-\lambda_{\text{wm}})\bar{V}_{s+1} + \lambda_{\text{wm}} y_{s+1}\big],$$

computed backward for $s = t+H-1, \dots, t$. Each gradient step recomputes and detaches targets using the current EMA encoder and value head, then updates online and EMA parameters (Algorithm 1). Rollouts start at $\hat{z}_t = \phi_\theta(o_t)$ and use $\hat{z}_{s+1} = f_\theta(\hat{z}_s, \bar{a}_s)$, evaluating heads at $\hat{z}_s$.

Define continuation masks $M_{t,0} = 1$ and $M_{t,j+1} = M_{t,j}(1-d_{t+j})$. To express all losses with the same normalization, define the masked average of a per-transition loss sequence ℓ as

$$\langle \ell \rangle_\eta = \mathbb{E}_{\mathcal{B}_k}\left[\frac{\sum_{j=0}^{H-1} M_{t,j+\eta}\ell_{t+j}}{\sum_{j=0}^{H-1} M_{t,j+\eta} + \epsilon_{\text{mask}}}\right], \qquad \eta \in \{0,1\}.$$

Using per-transition errors

$$\ell_{z,t} = \|\hat{z}_{t+1} - \text{sg}(\bar{z}_{t+1})\|_2^2,$$
$$\ell_{r,t} = \ell_r(\hat{r}_t, r_t), \qquad \ell_{v,t} = \ell_v(\hat{V}_t^{\text{ext}}, \text{sg}(y_t)),$$
$$\ell_{d,t} = \text{BCE}(\hat{d}_t, d_t),$$

where sg stops gradients, the complete objective is

$$\mathcal{L}_{\text{wm}} = \lambda_z \langle \ell_z \rangle_1 + \sum_{x \in \{r,v,d\}} \lambda_x \langle \ell_x \rangle_0. \tag{1}$$

Reward, value, and termination losses include terminal transitions. Latent consistency excludes auto-reset boundaries; each segment is normalized by its valid terms.

C. Stochastic Latent-Space Planning

Stochastic MPPI scores actor-sampled latent rollouts and fits a Gaussian to their first-step pre-squash actions. During iteration k, θ_k remains fixed throughout collection and on-policy updates, with $z_t = \phi_{\theta_k}(o_t)$. The actor prior is $v \sim \mathcal{N}(\mu_\psi(z), \text{diag}\,\sigma_\psi^2(z))$, followed by $u = \tanh(v)$, using only valid coordinates.

1) Candidate rollouts and scoring: For each candidate $i = 1, \dots, N$, initialize $\hat{z}_t^{(i)} = z_t$ and sample $\xi_{i,h} \sim \mathcal{N}(0, I)$ for $h = 0, \dots, H_p - 1$. Rollouts use

$$v_{t+h}^{(i)} = \mu_\psi(\hat{z}_{t+h}^{(i)}) + \sigma_\psi(\hat{z}_{t+h}^{(i)}) \odot \xi_{i,h},$$
$$\bar{a}_{t+h}^{(i)} = P_a^e(\tanh(v_{t+h}^{(i)})) \odot m_a^e,$$
$$\hat{z}_{t+h+1}^{(i)} = f_{\theta_k}(\hat{z}_{t+h}^{(i)}, \bar{a}_{t+h}^{(i)}).$$

Data collection uses $\psi = \psi_{\text{old}}$. Let $\hat{r}_h^{(i)}$, $\hat{d}_h^{(i)}$, and $\hat{V}_h^{(i)}$ denote reward, termination, and extrinsic value predictions along candidate i. Continuation probabilities satisfy $S_0^{(i)} = 1$ and $S_{h+1}^{(i)} = S_h^{(i)}(1 - \hat{d}_h^{(i)})$, giving

$$J_{\text{MB}}^{(i)} = \sum_{h=0}^{H_p-1} \gamma^h S_h^{(i)} \hat{r}_h^{(i)} + \gamma^{H_p} S_{H_p}^{(i)} \hat{V}_{H_p}^{(i)},$$
$$w_i = \frac{\exp(\kappa J_{\text{MB}}^{(i)})}{\sum_{j=1}^N \exp(\kappa J_{\text{MB}}^{(j)})}.$$

Here κ is an inverse temperature.

2) Planner policy and likelihood: The weighted first-step candidates define a diagonal Gaussian:

$$m_{v,t} = \sum_i w_i v_t^{(i)},$$
$$\Sigma_{v,t} = \text{diag}\left(\sum_i w_i (v_t^{(i)} - m_{v,t})^{\odot 2}\right) + \sigma_{\min}^2 I.$$

Here $\odot 2$ denotes elementwise squaring. The conditional planner distribution samples valid pre-squash actions as

$$v_t \mid z_t, \xi_t \sim \mathcal{N}(m_{v,t}, \Sigma_{v,t}),$$
$$u_t = \tanh(v_t). \tag{2}$$

Tokenizing and masking u_t gives the executed action. The conditional policy is $\tilde{\mu}_{\psi,\theta_k}^e(u_t \mid z_t, \xi_t)$, with candidate noise $\xi_t = \{\xi_{i,h}\}_{i,h}$ and implicit contexts.

Store ξ_t and executed v_t. On-policy updates recompute candidates, rollouts, scores, weights, and moments under ψ, holding (θ_k, z_t, ξ_t) fixed. The noise distribution and shared $\tanh$ Jacobian cancel from the likelihood ratio:

$$\rho_t(\psi) = \frac{\mathcal{N}(v_t; m_{v,t}(\psi), \Sigma_{v,t}(\psi))}{\mathcal{N}(v_t; m_{v,t}(\psi_{\text{old}}), \Sigma_{v,t}(\psi_{\text{old}}))}. \tag{3}$$

The ratio is evaluated using Gaussian log densities over valid coordinates. Gradients flow through the differentiable planner to ψ while θ_k remains fixed. Only the first action is executed; evaluation uses $\tanh(m_{v,t})$.

D. On-Policy Actor–Critic Learning

Following M3PO [3], M4PO combines on-policy actor–critic learning with value-discrepancy exploration. An extrinsic critic $V^{\text{ext}}_{\omega_{\text{ext}}}$ provides the model-free value estimate, while an augmented critic $V^{\text{aug}}_{\omega_{\text{aug}}}$ supplies policy advantages. Both are distinct from the world-model value head V^{ext}_{θ}.

Discrepancy bonus. For each executed transition, let $\hat{z}_{t+1} = f_{\theta_k}(z_t, \bar{a}_t)$ and $z_{t+1} = \phi_{\theta_k}(o_{t+1})$. Model-based and model-free one-step extrinsic estimates define

$$\begin{aligned} Q^{\text{ext}}_{\text{MB},t} &= \hat{r}_t + \gamma(1-\hat{d}_t)V^{\text{ext}}_{\theta_k}(\hat{z}_{t+1}), \\ Q^{\text{ext}}_{\text{MF},t} &= r_t + \gamma(1-d_t)V^{\text{ext}}_{\omega_{\text{ext}}}(z_{t+1}), \\ \delta_t &= \left|Q^{\text{ext}}_{\text{MB},t} - Q^{\text{ext}}_{\text{MF},t}\right|. \end{aligned} \tag{4}$$

Here $\hat{r}_t$ and $\hat{d}_t$ are evaluated at $(z_t, \bar{a}_t)$ using θ_k. We normalize and clip δ_t before augmenting the reward:

$$\begin{aligned} \epsilon_t &= \text{clip}\left(\frac{\delta_t - \mu_\delta}{\sigma_\delta + \epsilon_{\text{norm}}}, 0, \epsilon_{\max}\right), \\ \tilde{r}_t &= r_t + \beta\epsilon_t. \end{aligned}$$

Policy and value updates. Both critics use terminal-masked generalized advantage estimation (GAE) [16] with rollout-end bootstrapping: the extrinsic critic uses r_t, and the augmented critic uses $\tilde{r}_t$. Critic losses regress to the corresponding return targets, with rollout advantages and return targets held fixed throughout the PPO updates on that batch. PPO updates the stochastic planner-policy using augmented advantages and the planner likelihood ratio defined in Eq. (3). Gradients pass through planning while the world model, rollout latents, and sampled planner noise remain fixed. The world model is then updated on the same fresh rollout batch.

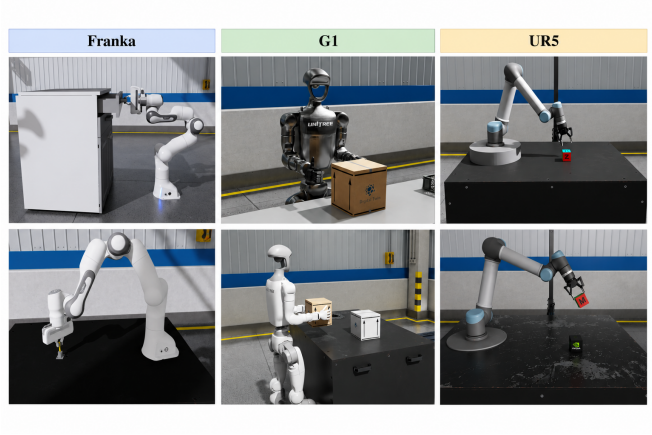


Fig. 3. Representative fixed-base, humanoid, and dexterous IsaacLab environments. Training randomizes textures, colors, lighting, and environment height.

Algorithm 1 Multi-Task Multi-Embodiment World-Model Update

Input: Batch $\mathcal{B}_k$, current $(\theta, \bar{\theta})$, horizon H, EMA decay ρ

for each gradient update **do**
  Sample a length-H segment and its (τ, e) from $\mathcal{B}_k$
  Compute contexts; set $\hat{z}_t \leftarrow \phi_\theta(o^e_t)$
  Encode and detach EMA targets $\bar{z}_{t+1:t+H}$
  for $j = 0, \ldots, H-1$ **do**
    Predict $(\hat{r}, \hat{V}^{\text{ext}}, \hat{d})_{t+j}$ from $(\hat{z}_{t+j}, \bar{a}^e_{t+j})$
    Roll out $\hat{z}_{t+j+1} \leftarrow f_\theta(\hat{z}_{t+j}, \bar{a}^e_{t+j})$
  end for
  Compute terminal masks, detached value targets, and loss (1)
  Update $\theta \leftarrow \theta - \alpha_{\text{wm}}\nabla_\theta\mathcal{L}_{\text{wm}}$
  EMA-update the target encoder and value head
end for

Algorithm 2 On-Policy Planning and Discrepancy Optimization

Input: $\theta, \psi, \omega_{\text{ext}}, \omega_{\text{aug}}; \mathcal{T}, \mathcal{E}; H_p, N, T_{\text{roll}}, \beta$

Initialize EMA encoder and value head from online parameters
for iteration $k = 1, 2, \ldots$ **do**
  Set $\theta_k \leftarrow \theta$, $\psi_{\text{old}} \leftarrow \psi$, $\mathcal{B}_k \leftarrow \emptyset$
  Sample (τ, e) for each environment; reset and compute contexts
  for $t = 0, \ldots, T_{\text{roll}} - 1$ **do**
    Encode $z_t \leftarrow \phi_{\theta_k}(o^e_t)$
    Plan $(v^e_t, \xi_t, \log\mu_{\text{old},t}) \leftarrow \text{SMPPI}_{\theta_k,\psi_{\text{old}}}(z_t)$
    Execute $\bar{a}^e_t = P^e_a(\tanh(v^e_t)) \odot m^e_a$ via denormalization and interpolation
    Observe (r_t, d_t, o^e_{t+1}); encode z_{t+1}
    Compute the clipped normalized discrepancy bonus ϵ_t
    Store transition, $z_t, v^e_t, \xi_t, \log\mu_{\text{old},t}, \epsilon_t$ in $\mathcal{B}_k$
  end for
  Compute augmented rewards and terminal-masked GAE targets for both critics
  With (θ_k, z_t, ξ_t) fixed, recompute planner likelihoods and update actor and critics
  Update $(\theta, \bar{\theta})$ on $\mathcal{B}_k$ with Algorithm 1; anneal β
end for

E. Training Procedure

Algorithms 1–2 alternate policy and model updates on fresh rollouts. Contexts and masks are implicit; rollout steps execute across all environments in parallel.

IV. Experiments

We evaluate M4PO in IsaacLab [17] across multiple robot embodiments, grouped into fixed-base, dexterous, and humanoid manipulation paradigms. Fixed-base and dexterous settings each cover grasping, object relocation, articulated manipulation, precision assembly, and multi-object manipulation; humanoid settings cover grasping and relocation, with results averaged over single-arm and bimanual modes. Results are summarized in 12 paradigm–category combinations across the evaluated robot embodiments. Figure 3 shows representative environments. Rewards follow IsaacLab task definitions,

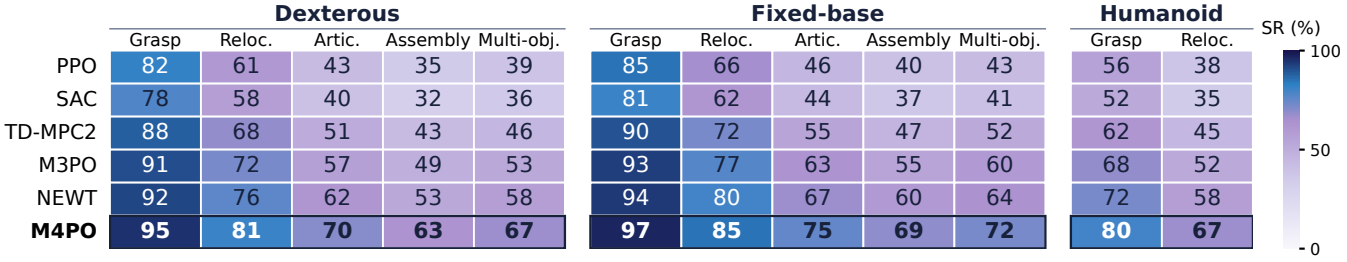


Fig. 4. Success rates (%) by paradigm and manipulation category. Humanoid values average single-arm and bimanual modes. Column labels abbreviate the categories in Sec. IV. Bold indicates the best method per condition; the M4PO row is outlined.

combining object progress, task completion, and control regularization.

Observations comprise RGB-D, proprioception, structured robot/task state, task language, and embodiment context (Figs. 1 and 2). No privileged simulator state is used. Variable-size state channels are padded and action channels tokenized; both are masked and normalized across embodiments, while external interpolation handles controller-rate differences. The humanoid uses the ASPIRE hand with actuated fingers in single-arm tasks and fixed fingers in bimanual tasks.

We distinguish single-task learning (ST), multi-task learning within a paradigm (MT), and multi-embodiment multi-task learning (ME-MT). The latter two share a policy across tasks and across both tasks and embodiments, respectively. We compare against PPO [18], SAC [19], TD-MPC2 [8], M3PO [3], and NEWT [9]. Results are averaged over three seeds and 100 evaluation episodes per task–embodiment pair. Metrics include success, episodic return, AUC, and worst-embodiment success where applicable. AUC uses success curves with the training axis normalized by aggregate environment steps. Across 2048 parallel environments, 100M aggregate steps correspond to approximately 48.8k high-level control steps per environment.

A. Simulation Across Manipulation Categories

Tables I–III report single-task success rates of M4PO and the baselines, using an independently trained policy for each task–embodiment setting. Results are grouped into 12 paradigm–category combinations across dexterous, fixed-base, and humanoid manipulation. Table IV reports multi-task training with a policy shared across tasks within each paradigm.

TABLE I

SINGLE-TASK SUCCESS RATES FOR DEXTEROUS MANIPULATION, ORGANIZED BY MANIPULATION CATEGORY. BEST RESULTS ARE BOLDED.

Method	Grasping	Relocation	Articulated	Assembly	Multi-object	Avg.
PPO	82.0	61.0	43.0	35.0	39.0	52.0
SAC	78.0	58.0	40.0	32.0	36.0	48.8
TD-MPC2	88.0	68.0	51.0	43.0	46.0	59.2
M3PO	91.0	72.0	57.0	49.0	53.0	64.4
NEWT	92.0	76.0	62.0	53.0	58.0	68.2
M4PO	**95.0**	**81.0**	**70.0**	**63.0**	**67.0**	**75.2**

TABLE II

SINGLE-TASK SUCCESS RATES FOR FIXED-BASE MANIPULATION, ORGANIZED BY MANIPULATION CATEGORY. BEST RESULTS ARE BOLDED.

Method	Grasping	Relocation	Articulated	Assembly	Multi-object	Avg.
PPO	85.0	66.0	46.0	40.0	43.0	56.0
SAC	81.0	62.0	44.0	37.0	41.0	53.0
TD-MPC2	90.0	72.0	55.0	47.0	52.0	63.2
M3PO	93.0	77.0	63.0	55.0	60.0	69.6
NEWT	94.0	80.0	67.0	60.0	64.0	73.0
M4PO	**97.0**	**85.0**	**75.0**	**69.0**	**72.0**	**79.6**

M4PO obtains the highest average success across all three manipulation paradigms. Gains are largest on contact-rich or coordination-heavy tasks such as articulated manipulation,

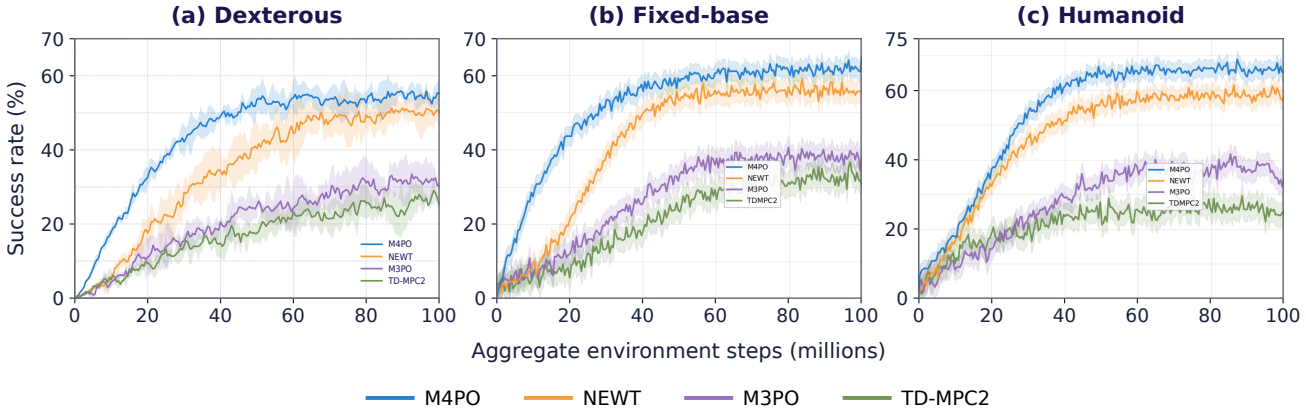


Fig. 5. Multi-task learning in (a) dexterous, (b) fixed-base, and (c) humanoid manipulation. Curves show measured success rates (%) versus aggregate environment steps (millions). Learning is non-monotonic: local dips persist after the initial improvement, and TD-MPC2 declines late in humanoid training. M4PO reaches the highest late-training success in all three settings.

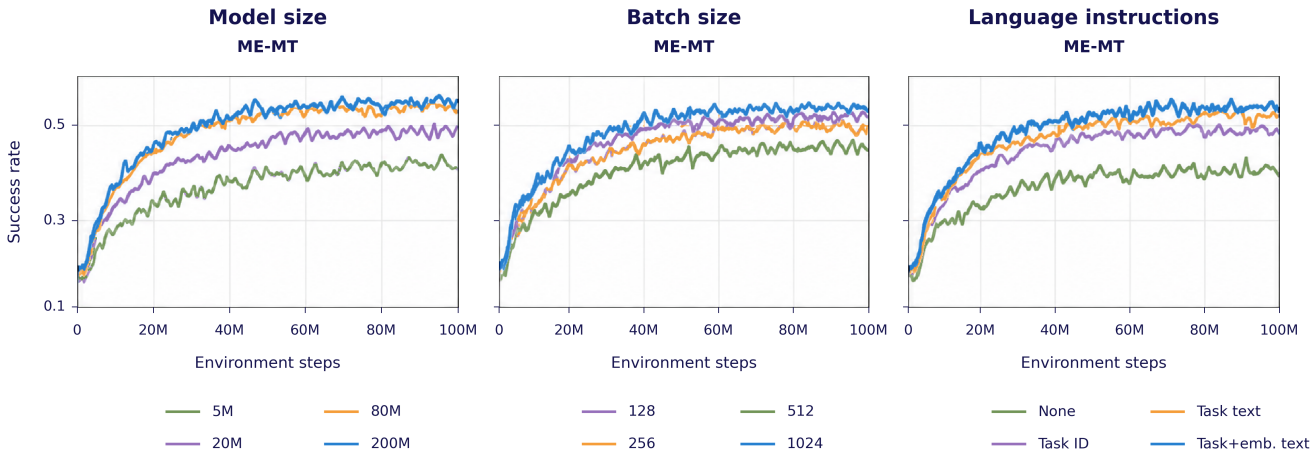


Fig. 6. Model-scale, batch-size, and instruction-conditioning ablations for M4PO in the ME-MT setting. Curves show success (0–1) versus environment steps, with local fluctuations throughout training. Left: gains diminish beyond 80M parameters; 80M and 200M reach similar late-training success. Middle: batch-size dependence is non-monotonic—1024 performs best, while 512 underperforms 128 and 256. Right: task plus embodiment text performs best, followed by task text, task IDs, and no instructions. These are within-method ablations; Fig. 7 compares the complete M4PO and NEWT systems.

TABLE III

SINGLE-TASK SUCCESS RATES FOR HUMANOID MANIPULATION, ORGANIZED BY MANIPULATION CATEGORY. EACH CATEGORY AVERAGES OVER SINGLE-ARM AND BIMANUAL VARIANTS.

Method	Grasping	Relocation	Avg.
PPO	56.0	38.0	47.0
SAC	52.0	35.0	43.5
TD-MPC2	62.0	45.0	53.5
M3PO	68.0	52.0	60.0
NEWT	72.0	58.0	65.0
M4PO	**80.0**	**67.0**	**73.5**

precision assembly, multi-object manipulation, and humanoid object relocation.

Figure 4 presents success rates across the 12 paradigm–category combinations.

B. Multi-Task Simulation Within Paradigms

The MT setting trains one policy across all tasks within each embodiment paradigm. Table IV reports final average success. Figure 5(a–c) shows measured learning curves for dexterous, fixed-base, and humanoid manipulation, respectively.

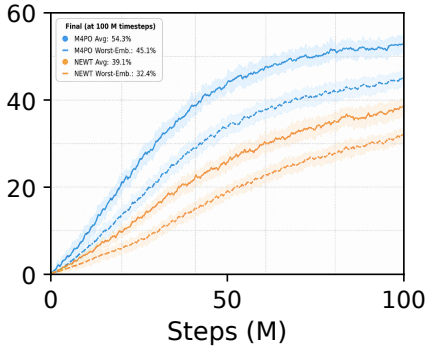


Fig. 7. Main ME-MT comparison of M4PO (blue) and NEWT (orange). Solid and dashed curves show average and worst-embodiment success (%), respectively; Fig. 6 varies design choices within M4PO.

TABLE IV

MULTI-TASK SUCCESS WITHIN EACH MANIPULATION PARADIGM.

Method	Dexterous	Fixed-base	Humanoid	Avg.
TD-MPC2	25.4	32.5	25.0	27.6
M3PO	30.2	38.5	36.0	34.9
NEWT	50.2	56.0	59.0	55.1
M4PO	**55.3**	**62.0**	**66.2**	**61.2**

TABLE V

FINAL MULTI-EMBODIMENT MULTI-TASK METRICS.

Metric	NEWT	M4PO
Avg. success	39.1	**54.3**
Avg. return	0.46	**0.62**
Worst-emb.	32.4	**45.1**
AUC	0.26	**0.37**

TABLE VI

RESULTS ON THE EXPANDED ARENA MULTI-EMBODIMENT MULTI-TASK SETTING.

Expanded Arena ME-MT	Avg. succ.	Worst emb.
NEWT	36.4	23.7
M4PO	**50.8**	**37.2**

M4PO improves over MT-NEWT across all three manipulation paradigms, with the largest gain of 7.2 percentage points on humanoid manipulation.

C. Multi-Embodiment Multi-Task Simulation

The ME-MT setting is the main benchmark: one policy is shared across the embodiments and tasks spanning fixed-base, dexterous, and humanoid manipulation, requiring transfer across morphologies, action subspaces, and manipulation modes. We compare against NEWT in Fig. 7 and Table V. We also evaluate the Isaac Lab-Arena benchmark [20] (Table VI).

TABLE VII

MATCHED ABLATIONS IN THE MULTI-EMBODIMENT MULTI-TASK SETTING. SUCCESS RATES ARE PERCENTAGES.

Variant	Avg. succ.	Worst emb.
Unfactorized shared latent	49.6	39.7
Factorized latent (M4PO)	**54.3**	**45.1**
Gaussian actor, same world model	46.9	36.8
Stochastic MPPI policy (M4PO)	**54.3**	**45.1**
Tokenized action, no learning mask	48.7	37.9
Shared masked action (M4PO)	**54.3**	**45.1**

M4PO improves average success by 15.2 percentage points and worst-embodiment success by 12.7 points over NEWT, indicating that gains are not limited to the easiest morphology. On the Isaac Lab-Arena benchmark, M4PO retains a 14.4/13.5-point average/worst-embodiment advantage over NEWT, supporting shared-policy learning on the expanded suite.

Table VII reports matched ME-MT ablations with the same observations, interaction budget, model capacity, training protocol, and three seeds; each comparison changes only the specified component.

Removing latent factorization, replacing the stochastic MPPI policy with a Gaussian actor, and removing action masking reduce average/worst-embodiment success by 4.7/5.4, 7.4/8.3, and 5.6/7.2 percentage points, respectively.

Model scale, batch size, and instruction conditioning. Figure 6 varies these design choices within M4PO in the ME-MT setting. Figure 7 compares the complete M4PO and NEWT systems, reporting both average and worst-embodiment success. Increasing model size from 5M to 80M improves success, while the 80M and 200M models attain similar late-training performance. A batch size of 1024 achieves the highest late-training success among the tested settings; the trend across the smaller batch sizes is non-monotonic. Conditioning on both task and embodiment text gives the highest late-training success, followed by task text, task-ID conditioning, and no instruction conditioning.

D. Real-World Experiments

We evaluate M4PO on Unitree G1 for blind RL transfer and Franka for language-conditioned fruit picking, sorting, and stacking (Figs. 8 and 9). Stacking includes both cube on cylinder and cylinder on cube. Each of the five conditions uses 20 hardware trials (100 total).



Fig. 8. Unitree G1 setup for blind RL sim-to-real transfer.

TABLE VIII

HARDWARE EVALUATION ACROSS FIVE TASK CONDITIONS, WITH 20 TRIALS PER CONDITION.

Task	Robot	Trials	Successes
Box lifting	G1	20	18
Fruit picking	Franka	20	19
Fruit sorting	Franka	20	15
Cube on cylinder	Franka	20	17
Cylinder on cube	Franka	20	16

TABLE IX

AVERAGE CPU TIME, WALL TIME, AND PEAK MEMORY BY REGIME. WALL-TIME ASSUME FULL USE OF 64 CPU THREADS; THEY ARE NOT ELAPSED-TIME MEASUREMENTS. RAM AGGREGATES PROCESS MEMORY.

Regime	Method	CPU time (h)	Est. min. wall (h)	GPU (GB)	RAM (GB)
ST	PPO	360	5.6	3.2	24
	SAC	470	7.3	4.1	66
	TD-MPC2	590	9.2	8.7	80
	M3PO	630	9.8	9.6	72
	NEWT	690	10.8	12.4	96
	M4PO	650	10.2	13.8	70
MT	TD-MPC2	910	14.2	12.2	112
	M3PO	960	15.0	13.0	104
	NEWT	1180	18.4	18.6	154
	M4PO	1040	16.3	20.4	118
ME-MT	NEWT	1520	23.8	25.8	212
	M4PO	1360	21.3	29.4	148

Training and transfer. The model is trained across simulated tasks, with additional G1-specific fine-tuning in simulation before deployment. Hardware evaluation uses fixed policy parameters without real-world policy training or fine-tuning, which we term zero-shot sim-to-real transfer. Fruit sorting is excluded from simulation training to test transfer to an unseen task composition.

Success criteria. G1 must lift the box without image observations. On Franka, picking requires visually identifying the requested fruit and holding it clear of the table; sorting requires placing all target fruits into two baskets as instructed while leaving the distractor untouched; stacking requires the instructed object order to remain stable after release and gripper withdrawal.

Evaluation metrics. For condition t, we report $\mathrm{SR}_t = 100\,k_t/20$, where k_t is the number of successful trials. The evaluation horizon $T_{\max}$ and required grasp/stack hold interval τ_{hold} are fixed before testing each condition. Timeouts, incorrect selections or stacking orders, failed grasps or placements, resets, and manual interventions count as failures.

E. Compute and Memory Usage

Table IX reports CPU time, minimum wall time, and peak memory.

Experiments used Ubuntu, an AMD Ryzen Threadripper 9970X (32 cores, 64 threads), an NVIDIA RTX 5090 (32 GB), and 128 GB RAM. CPU time sums simulator and learner execution across processes; provisional wall-time lower bounds use $T_{\mathrm{CPU}}/64$.

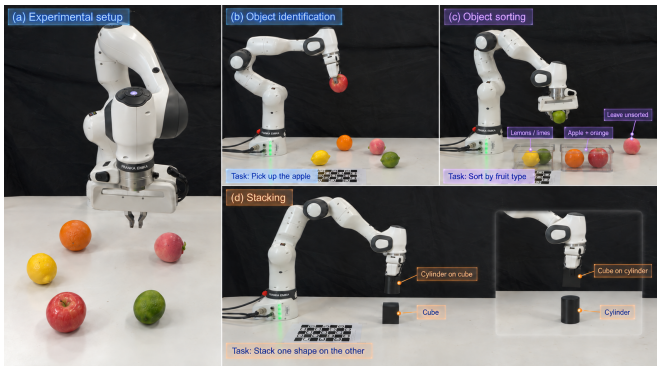


Fig. 9. Franka setup for language-conditioned fruit picking, sorting, and object stacking.

V. Conclusion and Limitations

We introduced M4PO, a multi-task, multi-embodiment framework combining hierarchical world modeling, action masking, and on-policy planning and learning. Across the evaluated simulation regimes, M4PO improves final average success over the baselines, including an improvement of 15.2 percentage points over NEWT on the main multi-embodiment benchmark. Hardware evaluation achieves 85% success across 100 trials on Unitree G1 and Franka, including 90% on G1 box lifting, without real-world policy updates.

Limitations include higher peak GPU memory and hardware validation on two robots across 5 conditions, with G1-specific adaptation in simulation and fixed G1 fingers during bimanual manipulation. The standardized action interface leaves low-level tracking, latency, calibration, and sim-to-real errors unresolved. Future work will expand robot and task coverage, enable articulated hand control, strengthen visual priors, and improve adaptation from limited hardware data.

References

[1] A. Narendra, M. Maribjonov, D. Makarov, D. Yudin, and A. Panov, "Knowledge-guided manipulation using multi-task reinforcement learning," 2026. [Online]. Available: https://arxiv.org/abs/2603.24083

[2] J. Luo *et al.*, "SERL: A software suite for sample-efficient robotic reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2024. [Online]. Available: https://arxiv.org/abs/2401.16013

[3] A. Narendra, D. Makarov, and A. Panov, "M3PO: Massively multi-task model-based policy optimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025, accepted at IROS 2025. [Online]. Available: https://arxiv.org/abs/2506.21782

[4] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," 2020. [Online]. Available: https://arxiv.org/abs/1912.01603

[5] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering Atari with discrete world models," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://arxiv.org/abs/2010.02193

[6] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse control tasks through world models," *Nature*, vol. 640, pp. 647–653, 2025. [Online]. Available: https://www.nature.com/articles/s41586-025-08744-2

[7] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 8387–8406. [Online]. Available: https://proceedings.mlr.press/v162/hansen22a.html

[8] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations*, 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html

[9] N. Hansen, H. Su, and X. Wang, "Learning massively multitask world models for continuous control," in *International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=MPabX9LEds

[10] M. Assran *et al.*, "Self-supervised learning from images with a joint-embedding predictive architecture," 2023. [Online]. Available: https://arxiv.org/abs/2301.08243

[11] M. J. Kim *et al.*, "OpenVLA: An open-source vision-language-action model," 2024. [Online]. Available: https://arxiv.org/abs/2406.09246

[12] J. Bjorck *et al.*, "GR00T N1: An open foundation model for generalist humanoid robots," 2025. [Online]. Available: https://arxiv.org/abs/2503.14734

[13] K. Black *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," 2025. [Online]. Available: https://arxiv.org/abs/2504.16054

[14] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[15] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

[16] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations*, 2016. [Online]. Available: https://arxiv.org/abs/1506.02438

[17] M. Mittal *et al.*, "Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning," 2025. [Online]. Available: https://arxiv.org/abs/2511.04831

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/abs/1801.01290

[20] NVIDIA Isaac Lab-Arena Contributors, "Isaac Lab-Arena: Composable environment creation and policy evaluation for robotics," 2025. [Online]. Available: https://github.com/isaac-sim/IsaacLab-Arena