Amortizing Generative Guidance for Model-Based Reinforcement Learning

Xiangteng Zhang[1] **Guojian Zhan**[1] **Likun Wang**[1]
Jingliang Duan[3] **Yang Guan**[1,†] **Shengbo Eben Li**[1,2,†]
[1]School of Vehicle and Mobility, Tsinghua University [2]College of AI, Tsinghua University
[3]School of Mechanical Engineering, University of Science and Technology Beijing
[†]Corresponding author.

Abstract

Planner-guided model-based reinforcement learning (MBRL) has emerged as a powerful paradigm for continuous control, combining learned world models with online planning to achieve strong performance and sample efficiency. However, existing methods typically distill actions optimized by the planner into a unimodal Gaussian policy and reuse it as a proposal prior for subsequent planning, overlooking the fact that Model Predictive Path Integral (MPPI) planners can induce multimodal supervision over high-value actions. Compressing such supervision into a unimodal distribution can lead to mode averaging, limited proposal coverage, and unstable bootstrapped policy learning. To address this issue, we propose GeMP, a **Ge**nerative **M**ultimodal **P**lanning framework for MBRL. We theoretically characterize the multimodality of supervision provided by the planner, showing that finite-sample planning can provide a distribution of high-value actions rather than a single deterministic target. To capture this distribution without compromising training stability, GeMP jointly learns a multimodal flow policy through value-weighted supervision and retains an auxiliary Gaussian policy for stabilizing off-policy actor-critic learning. We further employ a heterogeneous proposal generator that combines candidate action sequences from the Gaussian policy, the flow policy, and random sampling to improve proposal coverage for MPPI refinement. Experiments on MyoSuite and DeepMind Control Suite show that GeMP improves control performance and training stability over competitive baselines.

1 Introduction

Model-based reinforcement learning (MBRL) has emerged as an effective paradigm for continuous control by learning a predictive world model and using it for planning or policy optimization [19, 9, 23]. Compared with model-free reinforcement learning methods [12, 6, 24, 3], MBRL can exploit learned dynamics to predict future outcomes, evaluate action consequences, and provide additional model-based guidance for policy improvement, either through synthetic data generation [30, 31, 16], imagined rollouts [9, 8], or online planning [26, 11, 10, 37]. Among these directions, planner-guided MBRL is particularly appealing because it combines the local improvement capability of online planning with the sample efficiency of off-policy policy learning, leading to strong performance in continuous control.

Existing planner-guided MBRL methods commonly adopt Model Predictive Path Integral (MPPI) [35] as the core mechanism for online action selection, where candidate action sequences are sampled, evaluated, and refined in the learned world model, and the resulting optimized actions are used for policy learning [11, 10, 37]. These methods often further use the learned policy as a proposal

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

prior for subsequent planning, forming a closed bootstrapping loop between planning and policy learning [33, 37].

However, most existing methods implicitly treat the planner output as a deterministic point target for a given state, and train a Gaussian policy to fit the actions returned by the planner. This view overlooks the distributional structure induced by finite-sample planning. For the same latent state, a single MPPI execution returns a deterministic solution once its sampled candidates, perturbations, and final selection are fixed, but repeated executions can select actions from multiple separated high-return regions due to stochastic initialization and finite-sample refinement. As a result, MPPI can induce a multimodal conditional action distribution. Approximating this distribution with a unimodal Gaussian policy introduces a structural mismatch: the policy may average across separated high-value modes and weaken the behavioral diversity contained in planner supervision. Meanwhile, when reused as an MPPI proposal prior, an overly simple Gaussian initialization may also restrict the coverage of trajectory refinement and bias planning toward local modes [29, 25, 18]. This raises a central challenge for planner-guided MBRL: how can we model the multimodal distribution of planner outputs within a closed-loop learning framework while maintaining stable value learning?

Generative policies provide a natural way to model such complex supervision. Recent diffusion- and flow-based reinforcement learning methods have shown that expressive generative policies can represent multimodal action distributions beyond the capacity of conventional Gaussian actors [34, 17, 39]. However, directly replacing the Gaussian actor with a high-capacity flow policy in online model-based RL can compromise training stability. Only optimizing the policy to maximize the critic's value estimates can concentrate probability mass around currently favored value regions and erode the multimodal structure that the flow policy is intended to preserve [17]. Moreover, the multi-step generation process of flow policies may amplify critic error and world model bias under off-policy actor updates, leading to unstable learning [39, 38]. Therefore, the key is not simply to introduce a more expressive actor, but to use it to model the multimodal distribution of planner outputs while preserving the stability of critic-based policy learning.

To address this challenge, we propose **GeMP**, a **Ge**nerative **M**ultimodal **P**lanning framework for model-based reinforcement learning. GeMP integrates an expressive flow matching policy into a stable actor-critic framework with a learned world model. The Gaussian policy is trained to maximize estimated action values and provides a stable source of proposals for MPPI, while the flow policy serves as a generative branch for modeling the multimodal distribution of actions optimized by MPPI. Instead of directly updating the flow policy to maximize estimated action values, GeMP trains it with a value-weighted flow matching objective, where the critic modulates the relative importance of the actions returned by the planner without forcing the generative policy to collapse toward a single mode assigned a high value by the critic. During environment interaction, a heterogeneous proposal generator initializes MPPI with candidate action sequences from the Gaussian policy, the flow policy, and random sampling. This forms a closed-loop learning paradigm: planning provides supervision weighted by estimated action values for policy learning, while the learned policies provide increasingly informative and diverse proposals for subsequent planning.

Our contributions are threefold. First, we provide a distributional perspective on supervision provided by finite-sample MPPI, showing that repeated planning from the same state can induce a multimodal conditional action distribution over high-value actions. This perspective reveals a structural mismatch in existing planner-guided MBRL methods, where multimodal planner supervision is typically compressed into unimodal Gaussian policy learning. Second, we propose GeMP, a generative multimodal planning framework for model-based RL that combines a Gaussian actor-critic backbone with value-weighted flow matching. The Gaussian policy supports stable off-policy value learning and provides Gaussian proposals for MPPI, while the flow policy explicitly models the multimodal distribution of planner outputs without directly relying on unstable updates that maximize estimated action values. Third, we design a heterogeneous proposal generator that integrates candidate action sequences from the Gaussian policy, the flow policy, and random sampling for MPPI refinement, improving proposal coverage and supporting joint improvement of the planner and policy. Extensive experiments on DeepMind Control Suite and MyoSuite demonstrate that GeMP improves control performance and training stability over competitive baselines, confirming the benefit of generative multimodal proposals for planner-guided MBRL.

2 Preliminaries

Model-Based Reinforcement Learning. Reinforcement learning (RL) is commonly formulated as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{P}(s' \mid s, a)$ is the transition distribution, $r(s, a)$ is the reward function, $\gamma \in (0, 1)$ is the discount factor and $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces [13]. In this work, we focus on continuous-control tasks with proprioceptive state observations and continuous action spaces. The goal is to learn a policy π that maximizes the expected discounted return $J(\pi) = \mathbb{E}_{\pi,\mathcal{P}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. Model-based reinforcement learning improves policy learning and decision making by learning a world model of the environment dynamics and rewards. In latent-space MBRL, the world model encodes observations into latent states z, predicts future latent transitions and rewards, and provides a compact state representation for downstream control. Accordingly, the policy $\pi(\cdot \mid z)$ and critic $Q(z, a)$ are defined over latent states.

MPPI Planner. MPPI control is a sampling-based planner for model predictive control. Given a latent state z_t, MPPI optimizes a finite-horizon action sequence by sampling candidate trajectories from a proposal distribution, evaluating their model-predicted returns, and refining the proposal according to reward-weighted trajectory scores. Starting from N candidate action sequences $\tau_i = (a_i^0, \ldots, a_i^{H-1})$, MPPI computes the cumulative return R_i of each sequence under the learned world model and assigns trajectory weights $\omega_i \propto \exp(R_i/\eta)$, where η is a temperature parameter controlling the sharpness of the weighting. After K refinement iterations, the planner executes the first action of the selected refined sequence, yielding a closed-loop model predictive control procedure.

Flow Matching Policy. Flow matching [15] defines a generative policy through a time-dependent velocity field that transports a simple noise distribution to a target action distribution conditioned on the current latent state. Given an initial noise sample $x^0 \sim \mathcal{N}(0, I)$ and a latent state z, the action is generated by integrating the ordinary differential equation $dx^t/dt = v_\theta(t, x^t, z)$ from $t = 0$ to $t = 1$, where v_θ is a learnable velocity field. For a target x^1 from a dataset $\mathcal{D}$, flow matching trains the velocity field along the linear interpolation path $x^t = (1 - t)x^0 + tx^1$ by minimizing the conditional flow matching loss, defined as the squared error between $v_\theta(t, x^t, z)$ and a constant velocity field $x^1 - x^0$. The resulting flow policy, denoted by $\pi_\theta^f(\cdot \mid z)$, can be viewed as a stochastic mapping from Gaussian noise to actions conditioned on the latent state z. In practice, samples from π_θ^f are obtained by numerically integrating the learned velocity field, commonly using discretization schemes such as the Euler or midpoint method.

3 Method

3.1 Multimodality in planning supervision

Despite the successful application of MPPI in MBRL methods such as TD-MPC [10, 11] and BOOM [37], these methods often overlook the stochasticity of planner outputs at a given state. This is attributed to stochastic initial proposals, a finite number of optimization iterations, and random perturbations. In this section, we analyze how finite-sample MPPI selects among separated regions of high-return actions and characterize the resulting distribution of planner outputs.

At latent state z, consider M separated regions of high-return actions $\{\mathcal{B}_\ell(z)\}_{\ell=1}^M$. After K MPPI refinement steps, each final candidate is associated with a region label and a model-predicted return, denoted by (C_i^K, R_i^K) for $i = 1, \ldots, N$, where $C_i^K \in \{1, \ldots, M\}$ identifies the region containing the candidate action. Let $\rho_\ell^K(z) = \mathbb{P}(C_i^K = \ell \mid z)$ denote the probability that a refined candidate lies in region $\mathcal{B}_\ell(z)$, and let $F_\ell^K(\cdot \mid z)$ be the return distribution conditioned on $C_i^K = \ell$, with density $f_\ell^K(\cdot \mid z)$. The marginal return distribution of a refined candidate is $F^K(r \mid z) = \sum_{j=1}^M \rho_j^K(z) F_j^K(r \mid z)$.

The following result characterizes the probability that finite-sample MPPI selects an action from each region.

Theorem 1 (Region selection probabilities in finite-sample planning)**.** *Fix a latent state z. Suppose that the final refined candidates $(C_i^K, R_i^K)_{i=1}^N$ are conditionally independent and identically distributed given z, and ties in return occur with probability zero. If MPPI selects*

$i^\star = \arg\max_{1\le i\le N} R_i^K$, *then the probability that the selected action lies in region* $\mathcal{B}_\ell(z)$ *is*

$$p_\ell^{N,K}(z) = \mathbb{P}(C_{i^\star}^K = \ell \mid z) = N\rho_\ell^K(z) \int f_\ell^K(r \mid z) \left[F^K(r \mid z)\right]^{N-1} dr.$$

Theorem 1 shows how the selection probability of each action region depends on four factors. The coverage probability $\rho_\ell^K(z)$ measures how likely refined candidates are to lie in region $\mathcal{B}_\ell(z)$. The conditional return distribution $F_\ell^K(\cdot \mid z)$ describes the returns of candidates in that region. The sample size N affects the distribution of the maximum candidate return, while the number of refinement iterations K shapes both the coverage of the action regions and their conditional return distributions.

Repeated planning from the same state with N candidate trajectories and K refinement iterations can assign nonzero probability mass to multiple high-return action regions. When the distribution of planner outputs is multimodal, fitting it with a unimodal Gaussian policy introduces a structural mismatch, as such fitting can average over separated high-value modes, motivating the use of a multimodal generative policy.

3.2 GeMP: Generative Multimodal Planning framework for model-based RL

To capture the multimodal distribution of planner outputs, we introduce a flow matching policy as an expressive generative policy for representing complex conditional action distributions. However, directly replacing conventional Gaussian actors with flow policies in off-policy model-based RL raises two key challenges: (1) **Mode collapse**: an objective that only maximizes estimated action values may concentrate probability mass in regions assigned high values by the critic, thereby eroding the multimodal structure that the flow policy is intended to preserve [17]. (2) **Training instability**: the multi-step generation process of flow policies can induce ill-conditioned gradients under off-policy actor updates, leading to inferior training stability [39, 38]. These challenges are further exacerbated in MBRL, where value estimates are affected by model bias and critic estimation error.

To address these challenges, we design a unified training framework for modeling the multimodal distribution of planner outputs. The critic is jointly trained with a conventional Gaussian policy $\pi_\phi^g(\cdot \mid z)$, which serves as a stable actor-critic backbone for policy improvement. In parallel, the flow policy $\pi_\theta^f(\cdot \mid z)$ is optimized with a value-weighted flow matching objective, where the learned critic assigns larger weights to actions optimized by MPPI that have high estimated values, enabling the flow policy to model this multimodal distribution. During planning, both learned policies are incorporated into a heterogeneous trajectory proposal generator, which combines Gaussian, flow-based, and random proposals to initialize MPPI.

Joint learning of world model and critic. GeMP learns a latent world model and a critic from replay data collected using the planner. Given an observation s, the encoder maps it to a latent state $z = h_\psi(s)$. The latent dynamics $f_\psi(z, a)$ predicts the next latent state, the reward head $R_\psi(z, a)$ predicts reward, and the critic $Q_\omega(z, a)$ estimates the action value. These components are trained jointly and provide the predictive rollouts and terminal values used by MPPI. Both R_ψ and Q_ω

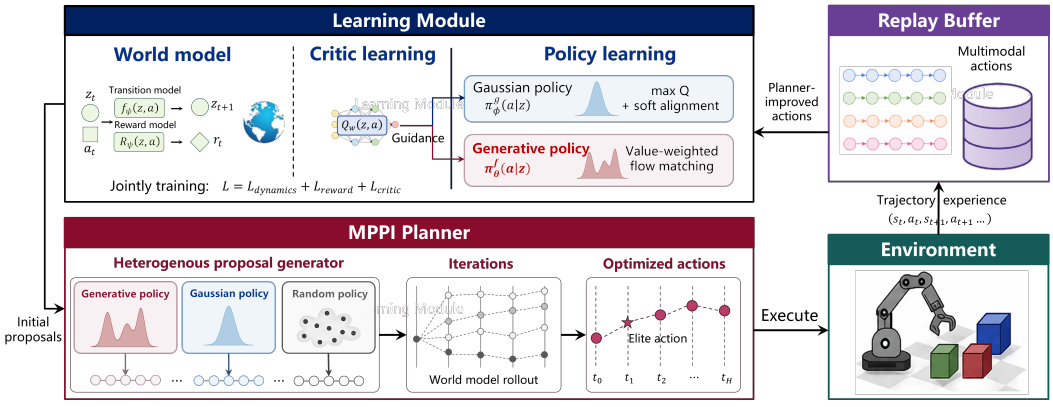


Figure 1: **GeMP**: Generative multimodal planning framework for model-based RL.

adopt a SymLog two-hot prediction, which is commonly used in recent algorithms [9, 10]. Denote $z_0 = h_\psi(s_0)$, $z_{i+1} = \hat{z}_{i+1} = f_\psi(z_i, a_i)$ along the model rollout, and $\bar{z}_{i+1} = h_\psi(s_{i+1})$. The loss of world model and critic is

$$\mathcal{L}_{\text{wm}} = \mathbb{E}_{\mathcal{D}}\left[\sum_{i=0}^{H-1} \rho^i \Big(\|\hat{z}_{i+1} - \text{sg}(\bar{z}_{i+1})\|_2^2 + \text{CE}(R_\psi(z_i, a_i), r_i) + \text{CE}(Q_\omega(z_i, a_i), \text{sg}(q_i)) \Big)\right], \quad (1)$$

where ρ is a temporal weighting factor, $\text{sg}(\cdot)$ denotes stop-gradient, CE denotes cross-entropy. The TD target is computed through the Gaussian actor-critic backbone: $q_i = r_i + \gamma \bar{Q}_{\bar{\omega}}(\bar{z}_{i+1}, \bar{a}_{i+1})$ with $\bar{a}_{i+1} \sim \pi_\phi^g(\cdot \mid \bar{z}_{i+1})$. The learned critic is then used as three roles: a terminal value estimator for MPPI, objective for Gaussian policy improvement, and a weighting signal for actions optimized by the planner in policy learning.

Value-weighted actor learning for planner supervision. GeMP maintains two policies with complementary roles: The Gaussian policy $\pi_\phi^g(\cdot \mid z)$ serves as a stable actor trained to maximize estimated action values. The flow policy $\pi_\theta^f(\cdot \mid z)$ is an expressive generative policy for modeling the distribution of planner outputs, so that multiple high-value action modes discovered by MPPI can be preserved. It should be noticed that since we consider a continuous action space, all the actions are normalized by performing $\tanh$ transformation.

The Gaussian policy is trained both to maximize estimated action values and to align with actions selected by the planner. Let $\tilde{a}_i$ denote the action optimized by the planner and stored in the replay buffer, and let $\hat{a}_i^g \sim \pi_\phi^g(\cdot \mid z_i)$. The Gaussian actor is optimized by

$$\mathcal{L}_g = \mathbb{E}_{\mathcal{D}}\left[\sum_{i=0}^{H-1} \rho^i \left(-Q_\omega(z_i, \hat{a}_i^g) + \lambda_g \bar{w}_i \left[-\log \pi_\phi^g(\tilde{a}_i \mid z_i) \right] \right)\right], \quad (2)$$

where λ_g controls the strength of this alignment and $\bar{w}_i = \exp(Q_i/\tau_{\text{align}}) / \sum_{j \in \mathcal{B}} \exp(Q_j/\tau_{\text{align}})$ over minibatch $\mathcal{B}$, with detached critic scores $Q_i = \text{sg}[Q_\omega(z_i, \tilde{a}_i)]$. The first term maximizes the critic value of Gaussian policy actions, while the second aligns the Gaussian actor with actions refined by the planner through weighted negative log-likelihood. Consistent with BOOM [37], higher-valued historical planner actions receive stronger imitation weight.

For the flow policy, we use value-weighted flow matching in the space before applying $\tanh$. Given a bounded action optimized by the planner, $a \in (-1, 1)^d$, we first compute $x^1 = \text{atanh}(a)$. We then sample a base sample $x^0 \sim p_0$ and a flow time $t \sim \mathcal{U}[0, 1]$, and define $x^t = (1-t)x^0 + tx^1$. The flow loss is denoted as

$$\mathcal{L}_f = \mathbb{E}_{\mathcal{D}, p_0, t}\left[w_i \left\| v_\theta(z_i, x_i^t, t) - (x_i^1 - x_i^0) \right\|_2^2 \right], \quad (3)$$

where $w_i = \text{sg}[\text{clip}(\exp(Q_\omega(z_i, a_i) - \bar{Q}_{\mathcal{B}}), w_{\min}, w_{\max})]$, and $\bar{Q}_{\mathcal{B}}$ is the mean critic value in the minibatch. The critic adjusts the relative contribution of planner actions through detached weights in the flow matching objective. The clipped exponential assigns unit weight to actions whose estimated values are at or above the minibatch mean and continuously downweights those below it. The positive lower bound retains supervision from every sample, while the upper bound prevents large value estimates from amplifying individual regression losses. Appendix C.1 examines the roles of graded weighting and clipping in flow matching.

Heterogeneous proposal generator for MPPI planning. During environment interaction, GeMP uses the learned policies as proposal generators for MPPI rather than directly executing their actions. Given the current latent state z, GeMP constructs a heterogeneous candidate set

$$\mathcal{T}^0 = \mathcal{T}^g \cup \mathcal{T}^f \cup \mathcal{T}^r, \quad (4)$$

where $\mathcal{T}^g$, $\mathcal{T}^f$, and $\mathcal{T}^r$ denote candidate action sequences generated by the Gaussian policy, the flow policy, and random sampling, respectively. Each trajectory uses one proposal source throughout its horizon; trajectories from the learned policies are generated autoregressively: at step j, an action is sampled from π_ϕ^g or π_θ^f, and the latent state is advanced by $z_{j+1} = f_\psi(z_j, a_j)$. This design combines stable exploitation from the Gaussian policy, multimodal coverage from the flow policy, and policy-independent exploration from random candidates.

Algorithm 1 GeMP: **Ge**nerative **M**ultimodal **P**lanning for Model-Based RL

Initialize world model $\hat{\mathcal{M}}_\psi$, critic Q_ω, Gaussian policy π_ϕ^g, flow policy π_θ^f, and replay buffer $\mathcal{D}$
for each training iteration **do**
  for each environment step **do**
    Encode observation o_t into latent state z_t
    Generate heterogeneous proposals $\mathcal{T}^0$ from π_ϕ^g, π_θ^f, and random sampling by (4)
    Run MPPI in $\hat{\mathcal{M}}_\psi$ to obtain a refined trajectory $\tilde{\tau}_t$
    Execute the first action $\tilde{a}_t$ and store the resulting experience in $\mathcal{D}$
  end for
  for each gradient step **do**
    Sample a minibatch from $\mathcal{D}$
    Update $\hat{\mathcal{M}}_\psi$ and Q_ω using (1)
    Update Gaussian policy π_ϕ^g using (2)
    Update flow policy π_θ^f using (3)
  end for
end for

MPPI evaluates and refines these candidates in the learned world model, executes the first action of a trajectory sampled from the elite set according to the return weights, and stores the resulting experience in the replay buffer. The replayed data are then used to update the world model, critic, Gaussian actor, and flow policy, forming a closed loop in which planning provides supervision weighted by estimated action values and the learned policies provide increasingly informative proposals for subsequent planning. MPPI remains active during evaluation; amortization refers to learning reusable proposals. The overall procedure is summarized in Algorithm 1.

3.3 Theoretical Analysis

We provide a theoretical analysis of how the value-weighted flow policy amortizes planner-induced improvement. At iteration k, let $\mu_k(\cdot \mid z)$ denote the proposal distribution before planning, $\pi_k^{\mathrm{plan}}(\cdot \mid z)$ the conditional distribution of planner actions in the replay buffer, and $\pi_{\theta_{k+1}}^f(\cdot \mid z)$ the updated flow policy after value-weighted flow matching. We denote replay samples by $(z_i, \tilde{a}_i)$, where $\tilde{a}_i$ is the first action returned by MPPI from latent state z_i. We define the planner target after value reweighting as

$$\bar{\pi}_k^{\mathrm{plan}}(a \mid z) \propto w_k(z,a)\, \pi_k^{\mathrm{plan}}(a \mid z),$$

where $w_k(z,a)$ is the critic-based weight used in the value-weighted flow matching loss. The following result states that, if the updated flow policy approximates this target sufficiently well, then it preserves planner-induced value improvement up to a controlled approximation error.

Proposition 1 (Monotone value reweighting)**.** *For a fixed critic $Q = Q_{\omega_k}(z,\cdot)$ and positive, non-decreasing $w(Q)$, we have*

$$\Delta_{\mathrm{rw}}(z) := \mathbb{E}_{\bar{\pi}_k^{\mathrm{plan}}}[Q] - \mathbb{E}_{\pi_k^{\mathrm{plan}}}[Q] = \frac{\mathrm{Cov}_{\pi_k^{\mathrm{plan}}}(Q, w(Q))}{\mathbb{E}_{\pi_k^{\mathrm{plan}}}[w(Q)]} \geq 0.$$

The clipped exponential weighting satisfies these conditions, so reweighting maintains or increases the expected critic value of the planner distribution. Proofs are detailed in Appendix D.

Theorem 2 (Preservation of value-weighted planner improvement under flow amortization)**.** *Assume that $Q_{\omega_k}(z,\cdot)$ is L_Q-Lipschitz on the compact action space $\mathcal{A} = [-1,1]^d$. Suppose that the planner distribution improves over the proposal distribution by at least $\Delta_{\mathrm{plan}}(z)$ under Q_{ω_k}. Set $\bar{\Delta}_k(z) = \Delta_{\mathrm{plan}}(z) + \Delta_{\mathrm{rw}}(z)$, and assume that the updated flow policy approximates this weighted target as $W_1\left(\pi_{\theta_{k+1}}^f, \bar{\pi}_k^{\mathrm{plan}}\right) \leq \varepsilon_k(z)$. Then when $\bar{\Delta}_k(z) > L_Q \varepsilon_k(z)$, the amortized flow policy has a higher expected critic value than the original proposal distribution:*

$$\mathbb{E}_{a \sim \pi_{\theta_{k+1}}^f}[Q_{\omega_k}(z,a)] \geq \mathbb{E}_{a \sim \mu_k}[Q_{\omega_k}(z,a)] + \bar{\Delta}_k(z) - L_Q \varepsilon_k(z).$$

Proof: See Appendix D.4.

Theorem 2 establishes that the flow policy retains a positive gain in expected critic value over the original proposal distribution when it approximates the reweighted planner distribution with sufficient accuracy. This result provides theoretical support for learning a generative policy from planner outputs through value-weighted flow matching and using the learned policy to generate proposals for MPPI.

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate our method on DeepMind Control Suite (DMC) [28] and MyoSuite [1], covering representative continuous-control tasks in locomotion and dexterous manipulation. We choose 7 challenging tasks in DMC, including Humanoid and Dog domains. In MyoSuite, we evaluate on ten hand-control tasks with both fixed and randomized goals, covering key turning, joint posing, fingertip reaching, object repositioning, and pen twirling.

Baselines. We compare our method against representative state-of-the-art reinforcement learning methods for continuous control. For model-free RL, we include (1) FlowRL [17]: a flow-based RL method with Wasserstein-2-regularized policy optimization, and (2) SAC [6]: a widely used off-policy algorithm based on maximum-entropy reinforcement learning. For model-based RL, we compare with (3) DreamerV3 [9]: an imagination-based MBRL method that trains policies on latent model rollouts, (4) TD-MPC2 [10]: a planning-based MBRL method that uses online MPPI planning for action selection, and (5) BOOM [37], which improves BMPC through a bootstrap loop that aligns the learned policy with behavior generated by the planner. Detailed descriptions are shown in Appendix A.3. For a fair comparison, we report results from the original papers whenever the corresponding benchmark results are available, and otherwise reproduce baselines using their official codebases.

4.2 Experimental results

As shown in Table 1, in locomotion tasks in DMC, GeMP achieves the highest average TAR of 841.7 over seven high-dimensional locomotion tasks. GeMP obtains the best performance on six out of seven tasks, while ranking second on Dog-stand. In manipulation tasks in MyoSuite, GeMP achieves the best average success rate of 0.86 over ten tasks. These results support our motivation that explicitly modeling the multimodal distribution of planner outputs is particularly beneficial for manipulation tasks, while also improving high-dimensional locomotion.

4.3 Visualization of Multimodality in Planner Supervision

To visualize the distribution of planner outputs, we fix a trained agent's latent state and repeatedly run MPPI with independent sampling randomness and warm-starting disabled. We collect the returned first actions and sample the flow and Gaussian policies at the same state. For each task, actions are projected onto a common coordinate, with mode intervals identified from planner samples. We select one task each from MyoSuite and DMC to visualize the separated planner modes captured by the learned multimodal policy, as shown in Figure 3. In both tasks, the planner density exhibits distinct peaks separated by a density valley, revealing multiple modes in the conditional action distribution induced by finite-sample planning at a fixed state. The flow policy assigns substantial probability mass to both planner mode intervals. The Gaussian policy concentrates most of its mass near one mode, leaving the other poorly covered. These examples illustrate how GeMP preserves multiple alternatives in planner supervision as proposals for subsequent MPPI refinement.

We also provide a handcrafted multi-goal navigation task in Appendix B, with equally rewarding goals arranged symmetrically around a fixed initial state. Repeated planning from this state produces distinct action modes associated with different goal directions. The flow policy captures these modes from planner supervision and generates successful trajectories toward each goal across repeated executions, preserving multiple solutions within a single conditional policy. The Gaussian policy concentrates its probability mass between goal directions and fails to reach a goal.

Table 1: Total Average Return (TAR) on 7 hard DMC tasks and Success Rate (SR) on 10 MyoSuite tasks. DMC tasks are evaluated at 2M environment steps, and MyoSuite tasks are evaluated at 1M environment steps. For each task, results are reported as mean ± std over 3 independent random seeds for fair comparison. **Bold** = best, underlined = second-best. Higher is better.

Task	SAC	FlowRL	DreamerV3	TD-MPC2	BOOM	GeMP (Ours)
Humanoid-stand	6.4 ± 0.1	864.5 ± 58.3	4.5 ± 0.5	663.7 ± 32.5	909.6 ± 16.8	**930.7 ± 13.3**
Humanoid-walk	2.0 ± 0.5	829.8 ± 74.2	1.6 ± 0.1	628.2 ± 16.3	901.4 ± 19.9	**927.9 ± 3.0**
Humanoid-run	1.2 ± 0.2	255.2 ± 51.6	0.8 ± 0.2	184.6 ± 23.7	443.0 ± 18.2	**528.2 ± 26.7**
Dog-stand	113.1 ± 84.4	**977.7 ± 4.2**	20.6 ± 1.8	798.9 ± 28.6	898.5 ± 109.2	966.6 ± 17.7
Dog-walk	15.3 ± 3.2	935.2 ± 5.0	6.1 ± 1.2	493.9 ± 381.3	952.1 ± 4.4	**960.5 ± 4.4**
Dog-trot	11.1 ± 3.3	885.8 ± 25.3	6.5 ± 1.4	500.0 ± 16.6	863.2 ± 46.6	**913.4 ± 13.8**
Dog-run	11.9 ± 5.3	516.8 ± 54.8	3.1 ± 2.2	169.9 ± 123.7	654.0 ± 36.1	**664.6 ± 23.3**
AVG. DMC Suite	23.0 ± 13.9	752.1 ± 39.1	6.2 ± 1.1	491.3 ± 89.0	803.1 ± 35.9	**841.7 ± 14.6**
Reach	0.67 ± 0.58	0.00 ± 0.00	0.00 ± 0.00	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
Reach-hard	0.27 ± 0.46	0.00 ± 0.00	0.00 ± 0.00	0.83 ± 0.15	0.77 ± 0.06	**0.97 ± 0.06**
Pose	0.67 ± 0.58	**1.00 ± 0.00**	0.00 ± 0.00	**1.00 ± 0.00**	0.95 ± 0.07	**1.00 ± 0.00**
Pose-hard	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
ObjHold	0.00 ± 0.00	0.87 ± 0.23	0.47 ± 0.21	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
ObjHold-hard	0.00 ± 0.00	0.27 ± 0.31	0.03 ± 0.06	0.57 ± 0.15	0.97 ± 0.06	**1.00 ± 0.00**
KeyTurn	0.00 ± 0.00	**1.00 ± 0.00**	0.07 ± 0.12	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
KeyTurn-hard	0.00 ± 0.00	0.20 ± 0.35	0.00 ± 0.00	0.00 ± 0.00	0.90 ± 0.00	**0.93 ± 0.06**
PenTwirl	0.00 ± 0.00	0.73 ± 0.12	0.00 ± 0.00	0.70 ± 0.52	0.00 ± 0.00	**1.00 ± 0.00**
PenTwirl-hard	0.00 ± 0.00	0.20 ± 0.20	0.00 ± 0.00	0.40 ± 0.00	0.40 ± 0.10	**0.70 ± 0.10**
AVG. MyoSuite	0.16 ± 0.16	0.43 ± 0.12	0.06 ± 0.04	0.65 ± 0.08	0.70 ± 0.03	**0.86 ± 0.02**

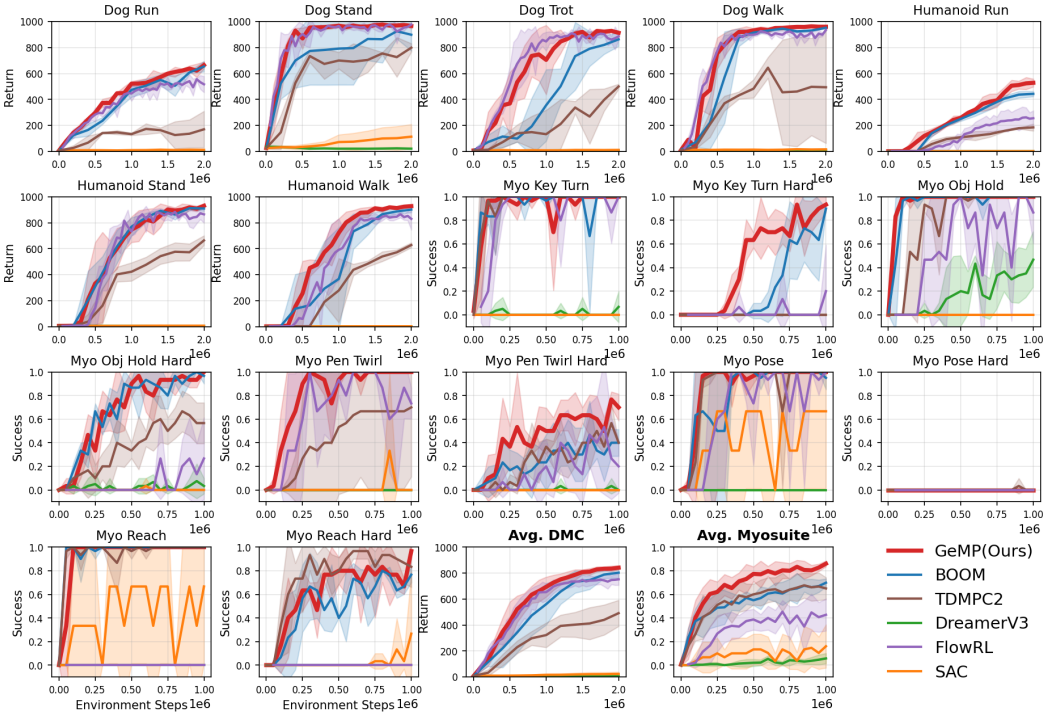
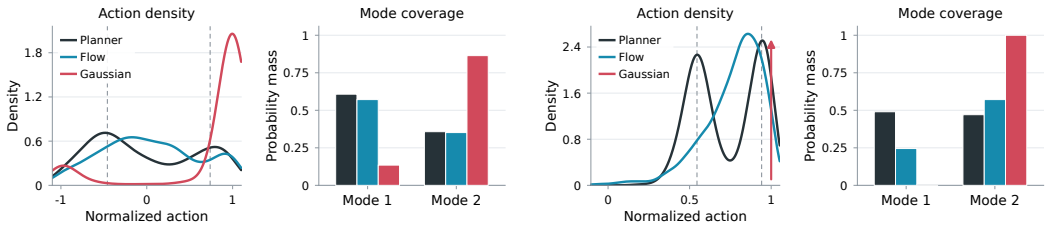


Figure 2: Results on 7 DMC Hard locomotion tasks and 10 Myosuite manipulation tasks. GeMP (Ours) achieves the highest average return and success rate. The solid lines are the mean value, while the shadow regions indicate the 95% confidence over 3 seeds.



(a) PenTwirl. (b) Humanoid-run.

Figure 3: Multimodal planner outputs and policy coverage at fixed states. The panels show action densities (left) and probability mass in planner mode intervals (right).

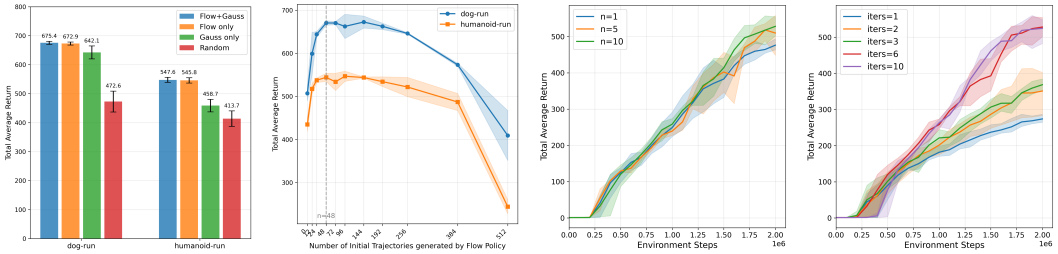
4.4 Ablation studies

Performance contribution of flow proposals. We first examine the necessity of learning a flow policy for MPPI initialization. At inference time, we compare MPPI with flow-only and Gaussian-only learned proposals, retaining the random proposal component. As shown in Fig. 4a, removing Gaussian proposals has only a minor effect on the final planner performance in Dog-Run and Humanoid-Run, whereas removing flow proposals lowers performance, particularly on Humanoid-Run. This suggests that flow proposals provide important initialization support for MPPI.

Number of flow proposals in initialization. We further study how the number of flow proposals among the 512 initial MPPI trajectories affects planner performance. As shown in Fig. 4b, a small number of flow proposals is sufficient to recover most of the performance gain, suggesting that the benefit comes from informative multimodal initialization. Allocating too many candidates to the flow policy can reduce the diversity of the overall proposal set and slightly weaken exploration.

Number of flow generation steps. We ablate the number of ODE integration steps used by the flow policy during action generation. As shown in Fig. 4c, using $n = 10$ flow generation steps achieves the best performance. Therefore, we use $n = 10$ in all reported results.

Number of MPPI refinement iterations. We also ablate the number of MPPI iterations used during planning, including 1, 2, 3, 6, and 10 iterations. As shown in Fig. 4d, increasing the number of refinement iterations allows MPPI to better explore the action space and improve upon the initial proposals, leading to higher returns.



(a) Ablation on effect of flow and Gaussian policies. (b) Ablation on number of flow proposals. (c) Ablation on different flow steps. (d) Ablation on number of iterations of MPPI.

Figure 4: Ablation studies and sensitivity analysis on locomotion tasks.

5 Related Works

Model-based RL. MBRL leverages a learned world model to facilitate efficient policy optimization and decision-making [27, 5]. Existing MBRL methods can be categorized into data augmentation, planning-driven and imagination-driven paradigms. Data augmentation methods [30, 31, 16] synthesize transitions or trajectories to expand the replay buffer. Imagination-based methods [7, 8, 9]

optimize policies and value functions on latent rollouts, improving sample efficiency but relying heavily on the accuracy of model prediction and its first-order derivative. Planning-based methods like the TD-MPC series instead use the world model in the environment interaction phase, where MPPI planner is applied for online optimization for short-horizon rollouts. BMPC further considers the actor divergence issue to ensure strict consistency between the planner and the policy [33]. BOOM introduces a closed bootstrap loop between policy alignment and online planning[37]. However, these methods predominantly rely on Gaussian policies for policy approximation, which remains a structural mismatch and can lead to mode averaging and collapse issues. In contrast, our method models this supervision as a multimodal conditional action distribution using a generative flow policy, facilitating consistency between the policy and planner and iterative improvement through planning and policy learning.

Generative policies for RL. Generative models have recently been explored as expressive policies [2, 36]. Diffusion policies can model multimodal and high-dimensional action distributions through value-guided policy improvement [34], entropy-regularized actor-critic learning [32], and policy-gradient fine-tuning [22]. Flow-based policies provide a deterministic ODE-based alternative to model complex action distributions [20], while recent online methods combine flow policies with Wasserstein-regularized improvement [17], velocity-reparameterized updates [39], entropy regularization [4], and likelihood-aware online fine-tuning [38]. Unlike these works, which mostly use the generative policy as the directly executed actor, our method trains the flow policy to approximate the multimodal distribution of MPPI outputs reweighted by estimated action values and uses it to provide diverse proposals for planner refinement. This value-weighted regression is related to advantage-weighted regression [21].

MPPI planning with generative policy. Model Predictive Path Integral (MPPI) control is a representative sampling-based MPC method that optimizes action sequences through importance-weighted trajectory sampling [35]. Recent studies emphasize that the proposal distribution is crucial for finite-sample MPPI. Trevisan et al. address the limited coverage of standard Gaussian sampling by fusing ancillary controllers and supporting arbitrary sampling distributions [29]. Li et al. connect MPPI, reinforcement learning, and diffusion models from a unified distributional refinement perspective [14]. CFM-MPPI trains an offline conditional flow prior for warm-started MPPI under prescribed dynamics [18], whereas GeMP updates its flow policy online from replay data collected using MPPI in a learned world model, with weights computed from a learned critic.

6 Conclusion

In this work, we revisit planner-guided model-based reinforcement learning from the perspective of supervision provided by the planner. Our analysis of how finite-sample MPPI selects among different high-return action regions provides a theoretical explanation for multimodal planning supervision. Motivated by this view, we propose GeMP, a generative multimodal planning framework and a heterogeneous proposal generator for MPPI refinement. The flow policy models the multimodal distribution of planner outputs, while the Gaussian policy provides a stable source of proposals through optimization of estimated action values and supports robust off-policy learning. GeMP improves proposal coverage and supports more effective joint improvement of the planner and policy. Experiments on MyoSuite and DeepMind Control Suite demonstrate that GeMP improves both control performance and training stability over competitive baselines. Appendix E discusses added cost, model dependence, and state-dependent benefits.

7 Acknowledgement

This work is supported by Tsinghua University - Toyota Joint Research Center for AI Technology of Automated Vehicle, and Tsinghua University (State Key Laboratory of Intelligent Green Vehicle and Mobility)–Xiaomi EV Technology Co., Ltd Joint Research Center for Autonomous Driving Safety and Intelligence.

References

[1] Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSuite – A contact-rich simulation suite for musculoskeletal motor control, May 2022. arxiv preprint. arxiv: 2205.13600.

[2] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, March 2024. arxiv preprint. arxiv: 2303.04137.

[3] Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, Shengbo Eben Li, Chang Liu, Ya-Qin Zhang, Bo Cheng, and Keqiang Li. Distributional Soft Actor-Critic With Three Refinements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3935–3946, May 2025.

[4] Ting Gao, Stavros Orfanoudakis, Nan Lin, Elvin Isufi, Winnie Daamen, and Serge Hoogendoorn. Flow Matching Policy with Entropy Regularization, March 2026. arxiv preprint. arxiv: 2603.17685.

[5] David Ha and Jürgen Schmidhuber. World Models, March 2018. arxiv preprint. arxiv: 1803.10122.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018. arxiv preprint. arxiv: 1801.01290.

[7] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination, March 2020. arxiv preprint. arxiv: 1912.01603.

[8] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models, February 2022. arxiv preprint. arxiv: 2010.02193.

[9] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025.

[10] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control, March 2024. arxiv preprint. arxiv: 2310.16828.

[11] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal Difference Learning for Model Predictive Control, July 2022. arxiv preprint. arxiv: 2203.04955.

[12] Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity Bias for Scaling Up Parameters in Deep Reinforcement Learning, May 2025. arxiv preprint. arxiv: 2410.09754.

[13] Shengbo Eben Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer, Singapore, 2023.

[14] Yankai Li and Mo Chen. Unifying Model Predictive Path Integral Control, Reinforcement Learning, and Diffusion Models for Optimal Control and Planning, March 2025. arxiv preprint. arxiv: 2502.20476.

[15] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[16] Cong Lu, Philip J. Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic Experience Replay, October 2023. arxiv preprint. arxiv: 2303.06614.

[17] Lei Lyu, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-Based Policy for Online Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[18] Kazuki Mizuta and Karen Leung. Unified Generation-Refinement Planning: Bridging Guided Flow Matching and Sampling-Based MPC for Social Navigation, March 2026. arxiv preprint. arxiv: 2508.01192.

[19] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based Reinforcement Learning: A Survey, 2020. arxiv preprint. arxiv: 2006.16712.

[20] Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning, May 2025. arxiv preprint. arxiv: 2502.02538.

[21] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

[22] Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization, December 2024. arxiv preprint. arxiv: 2409.00588.

[23] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arxiv preprint. arxiv: 1707.06347.

[25] Álvaro Serra-Gomez, Daniel Jarne Ornia, Dhruva Tirumala, and Thomas Moerland. A KL-regularization framework for learning to plan with adaptive priors, October 2025. arxiv preprint. arxiv: 2510.04280.

[26] Harshit Sikchi, Wenxuan Zhou, and David Held. Learning Off-Policy with Online Planning, October 2021. arxiv preprint. arxiv: 2008.10066.

[27] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, July 1991.

[28] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite, January 2018. arxiv preprint. arxiv: 1801.00690.

[29] Elia Trevisan and Javier Alonso-Mora. Biased-MPPI: Informing Sampling-Based Model Predictive Control by Fusing Ancillary Controllers. *IEEE Robotics and Automation Letters*, 9(6):5871–5878, June 2024. arxiv preprint. arxiv: 2401.09241.

[30] Likun Wang, Xiangteng Zhang, Yinuo Wang, Guojian Zhan, Wenxuan Wang, Haoyu Gao, Jingliang Duan, and Shengbo Eben Li. Off-policy Reinforcement Learning with Model-based Exploration Augmentation, October 2025. arxiv preprint. arxiv: 2510.25529.

[31] Renhao Wang, Kevin Frans, Pieter Abbeel, Sergey Levine, and Alexei A. Efros. Prioritized Generative Replay, October 2024. arxiv preprint. arxiv: 2410.18082.

[32] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, and Shengbo Eben Li. Diffusion Actor-Critic with Entropy Regulator, December 2024. arxiv preprint. arxiv: 2405.15177.

[33] Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped Model Predictive Control, April 2025. arxiv preprint. arxiv: 2503.18871.

[34] Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning, August 2023. arxiv preprint. arxiv: 2208.06193.

[35] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model Predictive Path Integral Control using Covariance Variable Importance Sampling, October 2015. arxiv preprint. arxiv: 1509.01149.

[36] Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yiheng Li, Yuxin Chen, Hongyang Li, Masayoshi Tomizuka, and Shengbo Eben Li. Mean Flow Policy with Instantaneous Velocity Constraint for One-step Action Generation, March 2026. arxiv preprint. arxiv: 2602.13810.

[37] Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap Off-policy with World Model. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[38] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning Flow Matching Policy with Online Reinforcement Learning, January 2026. arxiv preprint. arxiv: 2505.22094.

[39] Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC Flow: Sample-Efficient Reinforcement Learning of Flow-Based Policies via Velocity-Reparameterized Sequential Modeling, 2025. arxiv preprint. arxiv: 2509.25756.

A Experiment details

A.1 Benchmark settings

We provide a brief introduction to the environments used in our experiments. The selected benchmarks cover two complementary categories: high-dimensional whole-body locomotion from Deepmind Control Suite[28], and dexterous, contact-rich manipulation from MyoSuite[1]. Table 2 summarizes the observation and action dimensions of all evaluation tasks.

Table 2: MyoSuite /DMC Tasks: Observation and Action Dimensions

MyoSuite			**DMC**		
Task Name	**Obs Dim**	**Action Dim**	**Task Name**	**Obs Dim**	**Action Dim**
myo-reach	115	39	dog-run	223	38
myo-reach-hard	115	39	dog-stand	223	38
myo-pose	108	39	dog-trot	223	38
myo-pose-hard	108	39	dog-walk	223	38
myo-obj-hold	91	39	humanoid-run	67	21
myo-obj-hold-hard	91	39	humanoid-stand	67	21
myo-key-turn	93	39	humanoid-walk	67	21
myo-key-turn-hard	93	39			
myo-pen-twirl	83	39			
myo-pen-twirl-hard	83	39			

A.2 Environment introduction

For DMC tasks, we select seven tasks from the two most challenging domains: Humanoid and Dog, including three humanoid tasks and four dog tasks.



DMC Humanoid.

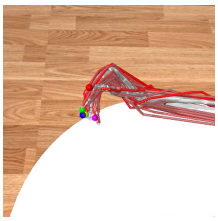
Tasks: `stand`, `walk`, and `run`.
Goal: Maintain upright balance, and move at target speeds of $1\,\mathrm{m/s}$ for walking and $10\,\mathrm{m/s}$ for running.
Reward: Standing reward for balance; moving rewards combine standing and forward-velocity terms.
Max steps: 1000.
Observation/action dim.: 67 / 21.



DMC Dog.

Tasks: `stand`, `walk`, `trot`, and `run`.
Goal: Maintain stable posture or generate quadruped gaits at $1\,\mathrm{m/s}$, $3\,\mathrm{m/s}$, and $9\,\mathrm{m/s}$.
Reward: Standing reward for support; moving rewards combine stability and forward-speed tracking.
Max steps: 1000.
Observation/action dim.: 223 / 38.

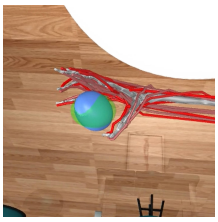
For MyoSuite, we evaluate five dexterous manipulation tasks based on the MyoHand model. Each task has two difficulty settings: Fixed (easy) and Random (hard). All tasks use a 39-dimensional muscle-actuation action space and a maximum episode length of 100 steps.



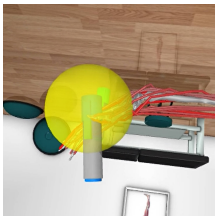
Hand Tips Reach.
Tasks: Fixed (easy) / Random (hard).
Goal: Move the fingertips to target spatial positions.
Reward: Negative fingertip-to-target distance with success bonuses and far-distance penalties.
Observation/action dim.: 115 / 39.



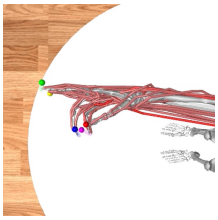
Hand Key Turn.
Tasks: Fixed (easy) / Random (hard).
Goal: Rotate a key to the desired angle using coordinated finger motions.
Reward: Key rotation, finger-key approach terms, action regularization, and success/failure bonuses.
Observation/action dim.: 93 / 39.



Hand Object Hold.
Tasks: Fixed (easy) / Random (hard).
Goal: Hold an object in the palm and move it toward the target without dropping it.
Reward: Negative object-to-target distance with success bonuses and drop penalties.
Observation/action dim.: 91 / 39.



Hand Pen Twirl.
Tasks: Fixed (easy) / Random (hard).
Goal: Rotate a pen to the target orientation while keeping it in hand.
Reward: Position and orientation alignment, action regularization, success bonuses, and drop penalties.
Observation/action dim.: 83 / 39.



Hand Tips Pose.
Tasks: Fixed (easy) / Random (hard).
Goal: Match target hand joint configurations.
Reward: Negative pose error with success bonuses, action regularization, and deviation penalties.
Observation/action dim.: 108 / 39.

A.3 Hyperparameters and Reproducibility

The parameters of GEMP across all tasks are summarized in Table 3. These hyperparameters are kept fixed to ensure a consistent and fair comparison. Our code implementation is built upon the BOOM codebase [37]. For fair comparison, GeMP and BOOM share the same world model architecture, critic architecture, replay buffer size, planning horizon, and total number of MPPI candidate trajectories. All experiments are conducted on a workstation equipped with two NVIDIA RTX 3090Ti GPUs, an Intel Core i9-12900K CPU, and 128GB RAM. The action repeat number is set to 2 in DMC environment and 1 in MyoSuite, which is consistent to TDMPC2 [10] settings. The result of DreamerV3, TDMPC, and TDMPC2 is directly from the original paper in TDMCP2.

Table 3: Hyperparameter Configuration.

Parameter	Value	Parameter	Value	Parameter	Value
Training					
Batch size	256	Learning rate	3×10^{-4}	Encoder LR scale	0.3
Optimizer	Adam				
Grad clip norm	20	Target net coef (τ)	0.01	Rho (ρ)	0.5
Discount min	0.95	Discount max	0.995	Discount denom	5
Buffer size	10^6	Agent decisions	1×10^6	Entropy coef	10^{-4}
Planning					
MPC iterations	6	Total samples	512	Num elites	64
Gaussian trajs	24	Flow trajs	48	Random trajs	440
Horizon	3	Min std	0.05	Max std	2.0
Inverse temperature	0.5				
Flow Matching					
Max flow weight	1.0	Min flow weight	0.001	ODE steps	10
Integrator	Midpoint	Flow hidden layers	2	Flow hidden width	512
Network Architecture					
Encoder dim	256	Num enc layers	2	MLP dim	512
Latent dim	512	Task dim	96	Num channels	32
Num Q networks	5	Num V networks	5	Dropout	0.01
SimNorm dim	8				
Distributional Critic and Reward					
Num bins	101	Vmin	-10	Vmax	+10
Policy					
Log std min	-10	Log std max	2	λ_g	$d/1000$
Alignment temp (τ_{align})	1				

A.4 Sensitivity to Random Seeds

To assess the sensitivity of GeMP to random seed selection, we extend the evaluation from three to five seeds per task. The suite averages in Table 4 remain close across both evaluations. We provide the code and per-run results in the supplementary material.

Table 4: Average GeMP performance over three and five seeds.

Benchmark	3 seeds	5 seeds
DMC (return)	840.2 ± 13.6	839.7 ± 11.4
MyoSuite (success rate)	0.86 ± 0.02	0.84 ± 0.04

B Analysis of Multimodal Supervision

We use a controlled four-goal navigation task to examine whether the flow policy preserves multiple useful action modes in planner supervision and converts them into successful behavior. Equally rewarding goals arranged symmetrically around a common initial state provide an interpretable connection between action modes and distinct task solutions.

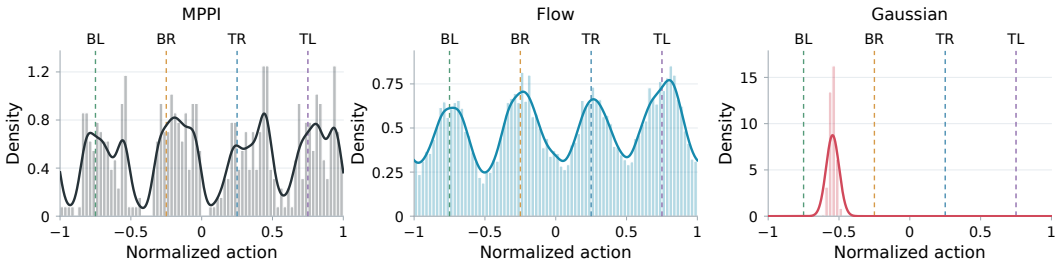
A robot starts at the origin and moves toward any of four goals at $(\pm 1, \pm 1)$. Its observation is its planar position $(x, y) \in [-1, 1]^2$. A scalar action $a \in [-1, 1]$ specifies the heading πa, with a step length of 0.1 and positions clipped to the workspace boundary. Reaching a goal within distance 0.1 yields a reward of 10 and terminates the episode. Otherwise, the reward is the negative distance from the resulting position to the nearest goal minus a 0.01 step cost. Episodes last at most 40 steps.

We train the agent online for $5{,}000$ interactions in one run, using MPPI outputs to supervise policy learning. This controlled setting uses a circular action representation for the flow policy and incorporates the task's fourfold rotational symmetry into replay, planning, and flow targets. We compare repeated MPPI outputs and samples from both learned policies at the origin, then evaluate

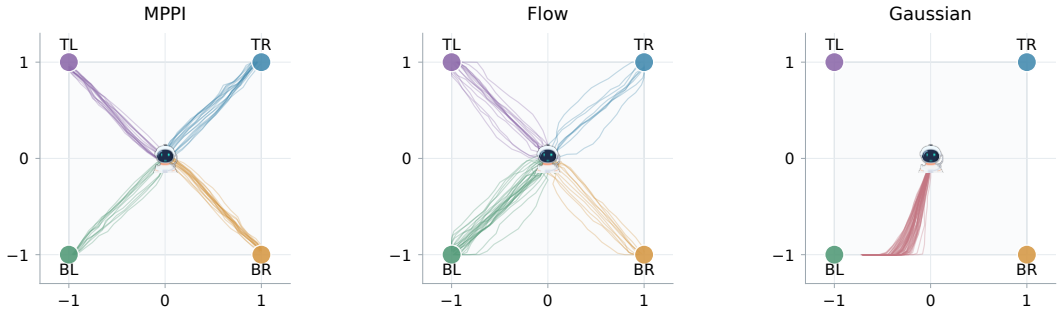
each controller over 64 episodes. The learned policies select actions directly during this evaluation. Table 5 summarizes direction coverage and execution performance, and Figure 5 shows the action distributions and resulting trajectories.

Table 5: Fixed-state and closed-loop diagnostics. Probability mass is measured within 0.12 of each target action, and a direction is covered when this mass is at least 5%. Episode goal counts use 64 episodes per controller and follow the TL/TR/BL/BR order.

Source	TL / TR / BL / BR (%)	Directions	Success	Episode goals	Return
Planner	15.6 / 12.7 / 15.0 / 16.2	4	100%	17 / 18 / 11 / 18	0.36
Flow	17.2 / 14.5 / 14.0 / 15.9	4	100%	17 / 9 / 27 / 11	-0.38
Gaussian	0 / 0 / 0.02 / 0	0	0%	0/0/0/0	-21.97



(a) Action distributions at the origin.



(b) Closed-loop trajectories from the origin over 64 episodes per controller.

Figure 5: Multimodal supervision in the four-goal navigation task. In (a), bars show normalized histograms and solid curves show circular kernel density estimates, with independently scaled density axes. Colored dashed lines mark the directions of the corresponding goals in (b), where the robot indicates the common starting position.

At the origin, MPPI assigns substantial probability mass around all four goal directions, and the flow policy retains four separated modes aligned with these directions. Both controllers achieve 100% success and reach all four goals across repeated episodes. The Gaussian policy concentrates between goal directions and fails to reach a goal in this evaluation. MPPI achieves the highest mean return, consistent with the benefit of continued planning during execution. These results demonstrate that the flow policy can preserve distinct successful behaviors from multimodal planner supervision within a single conditional policy.

C Additional Ablation Analyses

C.1 Value Weighting in Flow Matching

We examine graded value weighting and clipping in the flow matching objective on Humanoid-run. Let $\Delta Q_i = Q_\omega(z_i, a_i) - \bar{Q}_\mathcal{B}$ denote the critic value of a planner action relative to the minibatch mean. Clipped weighting (GeMP) clips $\exp(\Delta Q_i)$ to $[10^{-3}, 1]$. Unclipped weighting uses the same exponential with bounds of $[0, 10^3]$. Hard-threshold gating assigns $w_i = \mathbf{1}[\Delta Q_i \geq 0]$. Unweighted assigns unit weight to each sample.

At 2M environment steps, Clipped weighting (GeMP) reaches a mean return of 528.2 and Unclipped weighting reaches 528.1 (Fig. 6). Hard-threshold gating and Unweighted reach 492.3 and 499.4, respectively. Near the minibatch mean, a small change in the estimated value can switch a hard-threshold weight between zero and one. The clipped exponential varies continuously around this threshold and preserves a positive learning signal for planner actions below the minibatch mean.

Return variability under the default clipping bounds is lower at most intermediate checkpoints. At 1M environment steps, the standard deviations for Clipped weighting (GeMP) and Unclipped weighting are 14.0 and 84.0, respectively. The two variants achieve nearly identical final mean returns. Clipping helps stabilize training by substantially reducing across-run return variability at intermediate checkpoints.

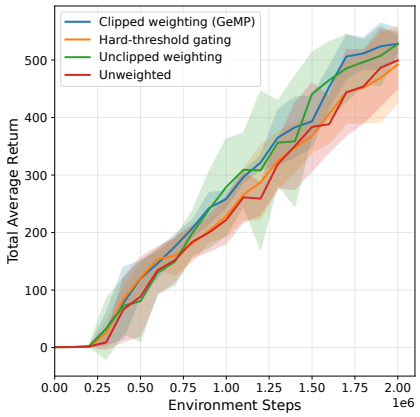


Figure 6: Learning curves for the four flow matching objectives on Humanoid-run. Solid lines show mean evaluation returns across three training runs per objective. Shading spans one sample standard deviation on either side of the mean.

C.2 Computational Cost

We measure end-to-end decision latency to quantify the additional inference cost of incorporating flow proposals into MPPI on single RTX 3090Ti GPU. We replace GeMP's flow proposals with Gaussian proposals while keeping the number of candidate trajectories and MPPI settings fixed. Table 6 shows that the default flow policy with 10 ODE steps increases latency from 27.64 to 37.31 ms, an overhead of 9.67 ms (35.0%). Reducing generation to one ODE step lowers the overhead to 1.57 ms (5.7%). The setting with 10 ODE steps also achieves the best control performance in the flow generation ablation (Fig. 4c), illustrating the trade-off between control performance and decision latency.

Table 6: Computational cost of incorporating flow proposals into MPPI.

Flow ODE steps	Gaussian (ms)	Gaussian+Flow (ms)	Overhead
1	27.67±0.86	29.24±0.73	1.57 ms (5.7%)
2	27.57±0.34	30.08±0.31	2.51 ms (9.1%)
5	27.54±0.29	32.76±0.35	5.23 ms (19.0%)
10	27.64±0.33	37.31±0.43	9.67 ms (35.0%)

D Proof of Theorems

D.1 Proof of Theorem 1

Fix z and omit the conditioning on z for notational simplicity. Under the assumptions of Theorem 1, MPPI selects the candidate with the largest return and ties occur with probability zero. Thus, the event

$\{C_{i^\star}^K = \ell\}$ can be decomposed over the N candidate indices. By symmetry among the conditionally independent and identically distributed refined candidates,

$$p_\ell^{N,K} = N\mathbb{P}\left(C_1^K = \ell,\, R_1^K \geq \max_{i=2,\dots,N} R_i^K\right).$$

Conditioning on $C_1^K = \ell$ and $R_1^K = r$, the probability that all remaining $N-1$ candidates have return no larger than r is

$$\mathbb{P}(R_2^K \leq r, \dots, R_N^K \leq r) = \left[F^K(r)\right]^{N-1},$$

where $F^K(r) = \sum_{j=1}^M \rho_j^K F_j^K(r)$ is the marginal return distribution of one refined candidate. Moreover, the joint probability of $C_1^K = \ell$ and $R_1^K \in dr$ is $\rho_\ell^K f_\ell^K(r)dr$. Integrating over all possible return values gives

$$p_\ell^{N,K} = N\rho_\ell^K \int f_\ell^K(r) \left[F^K(r)\right]^{N-1} dr.$$

Restoring the conditioning on z yields

$$p_\ell^{N,K}(z) = N\rho_\ell^K(z) \int f_\ell^K(r \mid z) \left[F^K(r \mid z)\right]^{N-1} dr,$$

which proves the theorem.

Relation to practical MPPI. Theorem 1 assumes independent and identically distributed refined candidates and selection of the candidate with the largest return. In GeMP, let $\mathcal{E}$ denote the final elite set and $s_i = \exp(\beta R_i)/\sum_{j\in\mathcal{E}} \exp(\beta R_j)$, with inverse temperature $\beta = 0.5$. Conditional on the complete candidate set $\mathcal{C}$, elite sampling gives

$$\mathbb{P}(C_{\text{selected}} = \ell \mid \mathcal{C}, z) = \sum_{i\in\mathcal{E}} s_i\, \mathbb{I}\{C_i = \ell\}.$$

Taking expectation over $\mathcal{C}$ gives the probability that the planner output lies in each action region. When the elite set covers multiple separated high-return action regions, its positive sampling weights assign a positive selection probability to each region, allowing repeated planning to return high-return actions from different regions.

D.2 Target of value-weighted flow matching

Fix z, a critic, and a positive weighting function $w(z,a)$. Let P_z be the conditional distribution of actions stored in replay after planning and exploration during data collection, and define $Z(z) = \int w(z,a)P_z(\mathrm{d}a) > 0$. The normalized target is

$$\bar{P}_z(\mathrm{d}a) = \frac{w(z,a)}{Z(z)} P_z(\mathrm{d}a). \tag{5}$$

For the squared error loss $\ell_\theta(x^0, a, t, z)$ used for velocity regression, independence of x^0, t from the endpoint weight gives

$$\mathbb{E}_{a\sim P_z, x^0, t}[w(z,a)\ell_\theta] = Z(z)\mathbb{E}_{a\sim \bar{P}_z, x^0, t}[\ell_\theta].$$

Thus, for each fixed state, weighting changes the endpoint distribution and multiplies the conditional objective by a positive constant independent of θ. Across states it also reweights their contributions by $Z(z)$; this matters under finite model capacity. With a realizable conditional velocity field, exact minimization of the population loss, and exact ODE integration, conditional flow matching transports the base distribution to the reweighted endpoint distribution [15]. Training in the space before applying $\tanh$ and then transforming the generated samples yields the corresponding distribution over bounded actions; in implementation, targets are clipped inside the boundary before the inverse transformation.

D.3 Proof of Proposition 1

Fix z and write $Z = \mathbb{E}_{\pi^{\text{plan}}}[w(Q)] > 0$. Then

$$\mathbb{E}_{\bar{\pi}^{\text{plan}}}[Q] - \mathbb{E}_{\pi^{\text{plan}}}[Q] = \frac{\mathbb{E}_{\pi^{\text{plan}}}[Qw(Q)]}{Z} - \mathbb{E}_{\pi^{\text{plan}}}[Q] = \frac{\mathrm{Cov}(Q, w(Q))}{Z}.$$

For independent copies Q, Q', $\mathrm{Cov}(Q, w(Q)) = \mathbb{E}[(Q-Q')(w(Q)-w(Q'))/2] \geq 0$ because w is non-decreasing, which proves the claim. Strictly positive weights also preserve the support of the original action distribution under reweighting.

D.4 Proof of Theorem 2

Proof. Fix a latent state z. For brevity, denote

$$P_f = \pi^f_{\theta_{k+1}}(\cdot \mid z), \qquad P_{\text{plan}} = \bar{\pi}^{\text{plan}}_k(\cdot \mid z), \qquad P_\mu = \mu_k(\cdot \mid z).$$

All three distributions are defined over the bounded action space $\mathcal{A} = [-1, 1]^d$. Although the flow model is trained in the coordinates $x = \operatorname{atanh}(a)$, the distribution P_f denotes the action distribution obtained after the transformation $a = \tanh(x)$. Since $\mathcal{A}$ is compact, all distributions have finite first moments and the Wasserstein distance W_1 is well-defined.

Let $f_z(a) = Q_{\omega_k}(z, a)$. By assumption, f_z is L_Q-Lipschitz on $\mathcal{A}$: $|f_z(a) - f_z(a')| \le L_Q \|a - a'\|_2, \forall a, a' \in \mathcal{A}$.

We first recall a standard consequence of the definition of the 1-Wasserstein distance. For any two probability distributions P and Q on $\mathcal{A}$, let $\Gamma(P, Q)$ denote the set of all couplings whose marginals are P and Q. Then

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \int_{\mathcal{A}\times\mathcal{A}} \|a - a'\|_2 \, d\gamma(a, a').$$

For any coupling $\gamma \in \Gamma(P, Q)$, because its first and second marginals are P and Q, respectively, we have

$$\mathbb{E}_{a\sim P}[f_z(a)] - \mathbb{E}_{a'\sim Q}[f_z(a')] = \int_{\mathcal{A}\times\mathcal{A}} \big(f_z(a) - f_z(a')\big) \, d\gamma(a, a').$$

Therefore, using the Lipschitz continuity of f_z,

$$\begin{aligned} |\mathbb{E}_{a\sim P}[f_z(a)] - \mathbb{E}_{a'\sim Q}[f_z(a')]| &\le \int_{\mathcal{A}\times\mathcal{A}} |f_z(a) - f_z(a')| \, d\gamma(a, a') \\ &\le L_Q \int_{\mathcal{A}\times\mathcal{A}} \|a - a'\|_2 \, d\gamma(a, a'). \end{aligned}$$

Since the above inequality holds for any coupling $\gamma \in \Gamma(P, Q)$, taking the infimum over all couplings gives

$$|\mathbb{E}_{a\sim P}[f_z(a)] - \mathbb{E}_{a\sim Q}[f_z(a)]| \le L_Q W_1(P, Q).$$

This inequality is the Lipschitz test-function bound underlying the Kantorovich–Rubinstein characterization of W_1.

Applying this bound with $P = P_f$ and $Q = P_{\text{plan}}$ yields

$$\mathbb{E}_{a\sim P_f}[Q_{\omega_k}(z, a)] \ge \mathbb{E}_{a\sim P_{\text{plan}}}[Q_{\omega_k}(z, a)] - L_Q W_1(P_f, P_{\text{plan}}).$$

Since $W_1(P_f, P_{\text{plan}}) \le \varepsilon_k(z)$, we obtain

$$\mathbb{E}_{a\sim P_f}[Q_{\omega_k}(z, a)] \ge \mathbb{E}_{a\sim P_{\text{plan}}}[Q_{\omega_k}(z, a)] - L_Q \varepsilon_k(z).$$

By Proposition 1 and the separately assumed planner gain, with $\bar{\Delta}_k(z) = \Delta_{\text{plan}}(z) + \Delta_{\text{rw}}(z)$,

$$\mathbb{E}_{a\sim P_{\text{plan}}}[Q_{\omega_k}(z, a)] \ge \mathbb{E}_{a\sim P_\mu}[Q_{\omega_k}(z, a)] + \bar{\Delta}_k(z).$$

Combining the last two inequalities gives

$$\mathbb{E}_{a\sim P_f}[Q_{\omega_k}(z, a)] \ge \mathbb{E}_{a\sim P_\mu}[Q_{\omega_k}(z, a)] + \bar{\Delta}_k(z) - L_Q \varepsilon_k(z).$$

Hence, when $\bar{\Delta}_k(z) > L_Q \varepsilon_k(z)$, the updated flow policy has strictly larger expected critic value than the original proposal distribution:

$$\mathbb{E}_{a\sim \pi^f_{\theta_{k+1}}(\cdot|z)}[Q_{\omega_k}(z, a)] > \mathbb{E}_{a\sim \mu_k(\cdot|z)}[Q_{\omega_k}(z, a)].$$

This completes the proof. □

E Limitations and Future Work

First, the flow policy brings additional computational overhead during training and proposal generation, particularly when multiple integration steps are used for action sampling. This may limit its applicability under strict real-time constraints. Second, the value-weighted flow supervision relies on the learned critic to prioritize actions optimized by the planner. Inaccurate value estimates caused by model bias or limited data may affect the quality of the weighting signal. Finally, GeMP uses MPPI as the refinement mechanism, whose finite number of samples and local perturbations may limit exploration across widely separated modes in complex environments. Future work may investigate more efficient flow sampling, uncertainty-aware weighting, and refinement mechanisms that are more directly aligned with multimodal proposal distributions.

F Positive and Negative Social Impact

GeMP improves planner-guided model-based reinforcement learning by better capturing the multimodal distribution of planner outputs and providing more diverse proposals for online planning. This may benefit real-world continuous control applications such as robotics, autonomous systems, and assistive devices, where sample efficiency, stable learning, and robust decision making are important. By leveraging learned world models and more informative planning proposals, GeMP may also reduce the number of costly physical interactions required during training. However, as with many reinforcement learning techniques, improved control and planning capability could be misused in sensitive domains and safety-critical systems. In addition, stronger simulation-based performance may encourage premature deployment before sufficient real-world validation. We therefore recommend careful evaluation, human oversight, and responsible use when applying such methods beyond controlled research environments.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim our main contributions, GeMP, a generative multimodal framework for model-based learning in abstract and explain the motivation clearly in introduction section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We summarize the limitations and future work of our research in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide thorough proof for all the theorems in Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We report the hyperparameters and implementation of our experiment in Appendix A.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide all the source code of our work with detailed README instructions in the supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We list all the hyperparmeters in Tab. 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars computed over independent runs with different random seeds, following a common practice in reinforcement learning experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the computing resources in Appendix A.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and complied with the NeurIPS Code of Ethics. The submitted code and supplementary materials are anonymized for double-blind review.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the positive and negative social impacts of our work in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: This paper builds on code, hyperparameters, and experimental settings from BOOM, FlowRL, and TD-MPC2. We properly cite the original works and acknowledge the corresponding assets used in our implementation and experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper introduces a new RL algorithm implemented in Python. We provide the source code and documentation in the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.