Generative Actor-Critic

Anonymous Author(s)
Affiliation
Address
email

Abstract

Conventional Reinforcement Learning (RL) algorithms, typically focused on estimating or maximizing expected returns, face challenges when online finetuning offline pretrained models. This paper introduces Generative Actor Critic (GAC), a novel framework that decouples sequential decision-making into two stages: learning a generative model of the joint distribution of trajectories and their returns, $p(\tau, y)$, and then performing decision-making via inference on this learned model. GAC offers a new perspective, framing *policy evaluation* as learning this comprehensive distribution, and enabling versatile *policy improvement* strategies through inference. To operationalize GAC, we introduce a specific instantiation based on a latent variable model that features a continuous latent plan vector. We develop novel inference strategies for both exploitation (optimizing latent plans for expected returns) and exploration (sampling latent plans conditioned on dynamically adjusted target returns). Experiments on Gym-MuJoCo benchmarks demonstrate GAC's strong offline performance and significantly enhanced offline-to-online adaptation compared to state-of-the-art methods, even in absence of stepwise rewards.

1 Introduction

A central objective in sequential decision-making is to *maximize the expected returns on trajectories* [1], denoted as $\mathbb{E}_{p(\tau)}[Y(\tau)]$, where each trajectory τ consists of a sequence of states and actions, and Y is a utility function assigning return y to each trajectory. Conventional Reinforcement Learning (RL) algorithms are designed to estimate and optimize this expectation during their training phase. The widely-adopted Actor-Critic framework [2, 3], for instance, exemplifies this by learning a *critic* to evaluate expected returns and an *actor* to refine the policy towards maximizing these returns. This expectation-centric paradigm is particularly well-suited for online learning settings common in traditional RL [1], due to its amenability to efficient, iterative updates from agent-environment interactions. However, in the context of modern Generative AI (GenAI) pipelines—often characterized by an extensive offline pre-training stage on vast datasets preceding online fine-tuning—we propose to move beyond this expectation-centric approach. We advocate for decoupling the decision modeling process into two distinct stages: (1) train-time generative modeling of the joint distribution over trajectories τ and their corresponding returns y:

$$p(\tau, y)$$

and (2) test-time decision-making, framed as an inference query based on this learned joint distribution. In this paper, we introduce this approach as the Generative Actor-Critic (GAC) framework.

The Generative Actor-Critic framework fundamentally shifts the paradigm from estimating trajectories' expected returns to learning a comprehensive distribution of trajectories and their returns. This holistic approach offers several key advantages. In particular, modeling the distribution $p(\tau, y)$ naturally entails a generative critic $p(y|\tau)$ and thus achieves *generalized policy evaluation*; this perspective underscores the ability of GAC to utilize data from various sources when modeling the

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

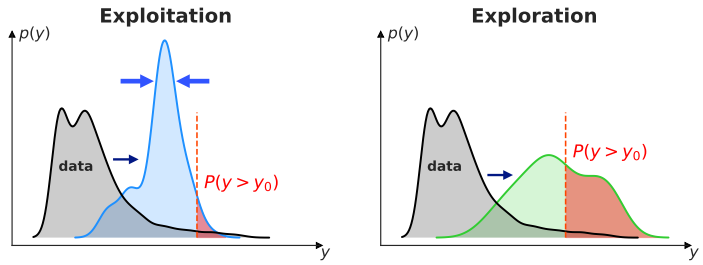

Figure 1: **Conceptual illustration of exploitation and exploration.** After modeling the data distribution (gray), GAC supports distinct test-time inference queries for various decision-making objectives. **(Left)** For *exploitation*, the objective is to maximize the expected return, leading to a focused, low-variance policy that targets high-certainty outcomes (blue). **(Right)** For *exploration*, the generative model is conditioned on a target distribution shifted towards higher returns, guiding the search for novel and potentially superior trajectories (green).

correlations between behaviors and outcomes, making GAC particularly well-suited for scenarios involving offline pre-training followed by online fine-tuning. Furthermore, by explicitly modeling the entire distribution, GAC can capture complex, multi-modal relationships between behaviors and outcomes—a capacity often limited in methods focusing solely on expectation. While this advantage is shared by distributional RL [4], the latter typically learns a return distribution conditioned on state-action pairs (often via a distributional Bellman equation) and subsequently falls back to the expectation of this learned return distribution for policy extraction [4–6]. In contrast, GAC directly models the joint distribution of trajectories and returns, enabling a fundamental shift that replaces expectation-based actor with a versatile inference process on $p(\tau|y)p(y)$. For instance, one could query the learned model to identify trajectories that maximize expected future returns, similar to policy optimization in distributional RL [5]. Alternatively, one could steer the generative process by conditioning on desired high returns to sample corresponding trajectories, akin to mechanisms in Decision Transformer (DT) [7, 8] and Diffuser [9, 10]. One could also sample diverse yet high-performing trajectories [11] to achieve *generalized policy improvement*. As we will demonstrate, such test-time flexibility can be leveraged, particularly in the offline-to-online setting, to address aspects of the classical exploration-exploitation dilemma. Fig. 1 illustrates the intuition. Starting from the data distribution, *exploitation* is to refine the policy to focus on a narrow distribution of high-certainty returns, while *exploration* is to actively seek novel, potentially superior outcomes by shifting the target return distribution towards uncharted, higher-value regions.

To operationalize the Generative Actor Critic framework and realize its potential, particularly for complex decision-making tasks and effective offline-to-online adaptation, this paper introduces several core methodological contributions: First, we instantiate GAC using a latent-variable model structured as $p(\tau, y, z) = p(\tau|z)p(y|z)p(z)$. Inspired by recent advances like Latent Plan Transformer (LPT) [12], a continuous latent plan z is introduced to effectively capture the correlation between high-dimensional trajectories τ and their corresponding low-dimensional returns y. Building on LPT's demonstration of efficient inference via such a latent space, we introduce non-trivial architectural and algorithmic modifications to significantly boost offline performance. Second, leveraging this latent structure, we design designated inference queries for exploitation and exploration. Exploitation is formulated as maximizing the expected return $\mathbb{E}[y|z]$ by performing gradient ascent directly in the latent plan space. For exploration, we propose to sample z from the posterior $p(z|y^+)$ of target returns from $p(y^+)$. These targets are dynamically set by sampling high-performing base returns y from the empirical distribution $p_{\text{data}}(y)$ (i.e., a replay buffer's top-k) and adding a small target improvement Δy, thereby guiding exploration towards potentially superior outcomes. The small target improvement and the prioritized replay buffer [13, 14] offer a "trust region" [15] for online fine-tuning. Empirically, GAC achieves competitive performance on Gym-MuJoCo benchmarks. Our framework demonstrates strong offline learning capabilities and significantly enhanced adaptation in offline-to-online scenarios compared to state-of-the-art baselines, even when step-wise rewards are absent during parts of its operation.

2 Preliminaries

Actor-Critic The conventional decision-making process in RL is Markov Decision Process (MDP)[1], represented by a tuple $\mathcal{M} = \langle S, A, P, \pi, R, \rho, T \rangle$, comprising a state space S, action space A, transition function $P : S \times A \mapsto \Pi(S)$, policy $\pi : S \mapsto \Pi(A)$, reward function

$R : S \times A \mapsto \mathbb{R}$, initial state distribution $\rho : \Pi(S)$, and horizon T. The decision-making objective is to maximize $\mathbb{E}[y]$, *i.e.*, the expectation of *return* $y := \sum_{t=0}^{T} R(s_t, a_t)$ accumulated along the trajectories $\tau = [s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T]$. One of the most useful identities in this formulation is the Bellman equation: if we define $Q(s_t, a_t) := R(s_t, a_t) + \mathbb{E}_{P,\pi}[\sum_{k=1}^{T-t} R(s_{t+k}, a_{t+k})]$, we will have $Q(s_t, a_t) = R(s_t, a_t) + \mathbb{E}_{P,\pi}[Q(s_{t+1}, a_{t+1})]$. This Bellman equation, as well as its various variants [1, 16, 17, 3], provides a foundation for learning a value function Q for the expectation of *return-to-go* at each step $RTG_t := \sum_{k=0}^{T-t} R(s_{t+k}, a_{t+k})$. Subsequently, the policy $\pi(a_t|s_t)$ can be updated along an approximated gradient direction $\nabla_{a_t} Q(s_t, a_t)$. That gives us the Actor-Critic framework [2, 18], where the *critic* Q estimates $\mathbb{E}[RTG_t|s_t, a_t]$, and the *actor* π optimizes the expectation from the critic.

Distributional RL Bellemare et al. [4] proposed to shift from the expectation-centric Bellman equation to the distributional version, $RTG_t|s_t, a_t = R(s_t, a_t) + RTG_{t+1}|s_{t+1}, a_{t+1}$. The fundamental difference is that RTG_t is a random variable, while the previous object of interest, Q_t, is its mean. Formally speaking, in distributional RL we model $p(RTG_t|s_t, a_t)$ when training the critic, and only calculate the expectation $\mathbb{E}[RTG_t|s_t, a_t]$ when extracting policy for the actor [4–6]. Bellemare et al. [4] showed that such a generative critic could express the multi-modality of $p(RTG_t|s_t, a_t)$ caused by *state aliasing* [19], *i.e.*, ambiguity of hidden states under insufficient memory context. However, Bellemare et al. [4] also formally derived that greedy policy selection (*i.e.* policy optimization toward $\nabla_{a_t} \mathbb{E}[RTG_t|s_t, a_t] = 0$) leads to unstable value iteration and does not guarantee contraction.

RL via Sequence Modeling Denoting the policy optimization result as $p(a_t|s_t, Q(s_t, a_t) = Q^*_{s_t})$, where Q^* is the optimal value, we obtain a generative perspective for the actor. This is what underlies the idea of return-conditioned behavior cloning [20, 7, 21]. Among them, Decision Transformer [7] proposed a generalized form of policy $\pi(a_t|s_{\leqslant t}, a_{<t}, RTG_{\leqslant t})$ that is learned with offline data similar to the autoregressive training of language models. However, in test time, since the model does not have access to the RTGs anymore, RTG_0 is set to be a particular high value (*e.g.*, the maximum or a quantile above the mean) in the offline data, and is updated as $RTG_{t+1} = RTG_t - r_t$ after observing rewards. To set more plausible targets, Lee et al. [11] proposed to model not only the action, but also the rewards and RTGs. This connects the generative actor $\pi(a_t|s_{\leqslant t}, a_{<t}, RTG_{\leqslant t})$ with the generative critic $p(RTG_t|s_{\leqslant t}, a_{\leqslant t}, RTG_{<t})$ in distributional RL and is closely related to our GAC framework. The difference is that GAC does not assume step-wise rewards, which is argued in Latent Plan Transformer [12] as a more naturalistic setup that results in decision-making models that can do *trajectory stitching* [22]. We will pinpoint a principled synergy between the generative critic and the generative actor for online policy improvement.

Offline-to-Online RL Offline-to-online RL [23–30] aims to enhance the sample efficiency of online fine-tuning by leveraging offline pre-training. However, conventional expectation-centric approaches often face challenges during this transition, as their critics can become unreliable due to the action distribution mismatch between the static offline dataset and the evolving online policy [31]. Efforts to mitigate this mismatch [32, 30] can, in turn, inadvertently curtail the exploratory capabilities of the learned actor. Generative actors, such as Decision Transformer, can be fine-tuned using a replay buffer that combines offline and online experiences [8]; yet, they often lack stable and scalable mechanisms for targeted exploration in the online phase. In contrast, GAC addresses this gap by providing a distributional perspective for targeted exploration: $p(\tau|y^+)p(y^+)$. This capability stems from GAC's distinct approach of jointly modeling trajectories and their associated returns, which facilitates more principled and flexible exploration strategies during online interaction.

3 Generative Actor Critic (GAC)

GAC is a framework that separates generative decision-modeling and decision-making as inference. In Section 3.1, we introduce several probabilistic inference queries for different decision-making strategies, motivating the employment of latent-variable generative models for efficient inference. In Section 3.2, we introduce a learning algorithm for GAC based on a latent-variable model inspired by Latent Plan Transformer (LPT) [12]. In Section 3.3 we describe the interplay between the actor, the critic, and the replay buffer [13, 14] in realizing policy improvement in exploratory online finetuning.

3.1 Decision-Making as Inference

As GAC features a generative model of $p(\tau, y)$, decision-making can be framed as principled inference over it. Previous works [16, 33–36] in decision-making as inference mainly take a stepwise perspective, in which the policy can be viewed as amortized variational distribution $q_{y*}(a_t|s_t)$ for the ground-truth posterior $p(a_t|s_t, y = y^*)$, where y^* denotes the optimal return. However, this posterior, along with its generalized versions in DT [7, 8], Diffuser [9, 10], LPT [12], is believed to have an *optimism bias* [16, 34]. Consider the decision of whether to buy a lottery, where $y^* = \$10M$ but it is almost impossible to get. The decision from $p(a|s, y = y^*)$ would be "to buy", while we all know that the optimal decision is "not to buy". Ziebart [16] attempted to bypass this issue by structuring the variational distribution of $p(\tau|y = y^*)$ as $\prod_0^{T-1} q_{y*}(a_t|s_t)P(s_{t+1}|s_t, a_t)$, where $P(s_{t+1}|s_t, a_t)$ is the ground-truth world dynamics.

We want to advocate for a more general solution to optimism bias. The issue is not in the inference process, but in the inference objective. As we all know, the most intuitive decision-making objective is to maximize the expected return $\mathbb{E}[y|\tau]$. And the actual optimal decision "not to buy" is optimal under this objective. Obviously, to a generative model $p(\tau, y)$, $\max_\tau \mathbb{E}[y|\tau]$ is a inference query distinct from $p(\tau|y = y^*)$. In fact, the logic underlying the decision policy $p(\tau|y = y^*)$ in the context of lottery is "If you want to win a lottery, you must first buy a lottery."

Solving $\max_\tau \mathbb{E}[y|\tau]$ directly would employ gradient ascent that requires costly backpropagation through time. This is where latent modeling excels. Consider a generative model with continuous latent variables z

$$p_\theta(\tau, y, z) = p_\alpha(z)p_\beta(\tau|z)p_\gamma(y|z), \tag{1}$$

where conditional independence is assumed between the trajectory τ and its return y, positioning z as an information bottleneck. Intuitively, the latent z are *plans* that abstract trajectories around their returns. As long as $p_\gamma(y|z)$ and $\mathbb{E}[y|z]$ have analytical forms, gradient-based inference can be lifted to the variational posterior $q_\phi(z)$ in the continuous latent space:

$$\max_\phi \mathbb{E}_{q_\phi(z)} \left[\mathbb{E}[y|z]\right]. \tag{2}$$

Once z is inferred, the policy is entailed by $p_\beta(\tau|z)$. The efficiency gain over $\max_\tau \mathbb{E}[y|\tau]$ is apparent.

Continuous latent modeling also enables other inference queries that would not be imaginable otherwise. In fact, the optimization in Eq. (2) can be viewed as a special type of joint sampling over $p(z, y)$. Returning to the lottery example, a decision policy $p(z|y)$ would not be unreasonable if we had more nuanced control over optimism. In other words, if, instead of sampling from a fixed target y^*, we sample from a marginal distribution $p(y^+)$ that is slightly shifted right from $p(y)$, the *optimism* in $p_\theta(z|y^+)$ is tamed with the awareness of $p(y^+)$. To sample from $p_\theta(z|y^+)$, we can employ classical Variational Bayes, which maximizes the evidence lower bound (ELBO) [37, 38]:

$$\max_\phi \mathbb{E}_{q_\phi(z)}[\log p_\gamma(y^+|z)] - D_{\mathrm{KL}}(q_\phi(z)||p_\alpha(z)). \tag{3}$$

This is equivalent to $\min_\phi D_{\mathrm{KL}}(q_\phi(z)||p_\theta(z|y^+))$, where $p_\theta(z|y) \propto p(z)p_\gamma(y|z)$. We will introduce how to sample from $p(y^+)$ in Section 3.3.

3.2 Generative Modeling

Latent Plan Transformer [12] is an instantion of the latent-variable generative model desribed in Eq. (1). Building on their conceptualization, we made two modifications that align well with some common sense about decision-making: (1) the initial state s_0 is not generated from a plan, so we move it into the conditioning; (2) the latent abstractions of the trajectories and their returns are better to be distinct and associated instead of identical, so we reserve one vector z_y for returns, leave the rest $z_{\setminus y}$ for trajectories, and associate them in the prior $p_\alpha(z)$. This gives us the factorization:

$$p_\theta(\tau, y, z) = \rho(s_0)p_\alpha(z|s_0)p_\beta(\tau|s_0, z_{\setminus y})p_\gamma(y|z_y). \tag{4}$$

Following LPT, the prior $p_\alpha(z)$ is implemented as an implicit generative model, but we replace the UNet transformation [39] in LPT with a Transformer encoder with bidirectional mask. To sample $z \sim p_\alpha(z)$, we first sample a set of isotropic Gaussian $\epsilon \sim \mathcal{N}(0, I)$ and transform them with the Transformer $z = f_\alpha(s_0, \epsilon)$.

We also inherit LPT's trajectory generator as a z-conditioned autoregressive model with a finite context window of size M: $p_\beta(\tau|s_0, z_{\setminus y}) = \prod_{t=0}^{T-1} p_\beta(a_t|s_{\leqslant t}, a_{<t}, z_{\setminus y}) p_\beta(s_{t+1}|s_{t-M:t}, a_{t-M:t}, z_{\setminus y})$. This design forces the latent z to serve as global carriers of information, bridging temporal segments that would otherwise be disconnected due to the limited context. The generator is parameterized by a causal Transformer decoder that incorporates the latent z via cross-attention. The stepwise policy and transition distributions are modeled as Gaussians with learnable or fixed variances.

The return predictor models $p_\gamma(y|z_y)$ as a Gaussian $\mathcal{N}(\mu_\gamma(z_y), \sigma_y^2)$, where μ_γ is an MLP that predicts y from the special latent vector z_y. The variance σ_y^2 is either treated as a hyperparameter or learned jointly. Note that $p_\gamma(y|z_y)$ being a Gaussian would not constrain the capability of modeling multi-modal return distribution because $p(y|\tau) = \int p(y|z_y)p(z_y|\tau)\mathrm{d}z_y$ can be very expressive.

For a data point (τ, y), Maximum Likelihood Estimate (MLE) of latent-variable models is intractable. Instead, we introduce a variational distribution $q_\phi(\epsilon|\tau, y) = \mathcal{N}(\mu, \sigma^2)$, parameterized by $\phi = (\mu, \sigma)$, *i.e.*, the mean vector μ and a diagonal covariance matrix σ^2 [40, 41], and maximize the evidence lower bound (ELBO) [37, 38]:

$$ELBO(\theta, \phi) = \mathbb{E}_{q_\phi(\epsilon)}[\log p_\theta(\tau, y|s_0, \epsilon)] - D_{\mathrm{KL}}(q_\phi(\epsilon)||p(\epsilon)). \tag{5}$$

We use the re-parametrization trick [42] in $\mathbb{E}_q$. The training employs alternating update of the local parameters ϕ specific to each (τ, y) with classical Variational Bayes [38, 37] and the global parameters $\theta = (\alpha, \beta, \gamma)$ shared across all training data. Kong et al. [12] recently showed that this training scheme can be scaled up to GPT-2 scale language models. To further accelerate and stabilize training, we store the optimized $\phi^*_{(\tau,y)}$ for each trajectory-return pair (τ, y), which are used as initializations whenever the data point is redrawn to constitute the batch.

Inference queries in Eq. (2) and Eq. (3) can be adjusted accordingly to incorporate conditioning on s_0 and reparametrization with ϵ. Between them, Eq. (2) enables exploitation during test time, while Eq. (3) facilitates exploratory online fine-tuning.

3.3 Exploratory Online Fine-tuning

A key advantage of the GAC framework is that the training objective—maximizing the ELBO in Eq. (5)—remains consistent for both offline pre-training and online fine-tuning. The online phase focuses on progressively improving the generative model by collecting higher-quality trajectories and incorporating them into the training data. This is achieved through a principled exploration strategy followed by staged model updates.

Our exploration mechanism is designed to sample trajectories from an improved distribution by targeting returns slightly higher than what the model has previously achieved. Ideally, one might tilt the model's prior $p(z)$ to generate a desired $p(y^+)$. However, we adopt a simpler and more direct approach. We leverage a prioritized replay buffer, initialized with the offline dataset, which serves as an empirical return distribution $p_{\text{data}}(y)$. To generate a target y^+ from the target distribution $p(y^+)$, we first sample a high-performing return y from the top-k quantile of the replay buffer. We then add a small, positive increment Δy to it, creating an optimistic target $y^+ = y + \Delta y$. This target is used to condition the inference process as described in Eq. (3), yielding latent plans $z \sim p(z|y^+)$ that guide the agent to explore potentially out-of-distribution (OOD) yet superior trajectories.

While the returns collected by the actor $p(z|y)p_{\text{data}}(y)$ from the environment are generally consistent with the target y sampled from the empirical distribution, the introduction of Δy creates a mismatch between the target distribution $p(y^+)$ and the ground-truth return distribution. How to reliably set Δy to balance exploration with stability is an interesting research question that we leave for future work. In this work, we treat Δy as a manually tuned hyperparameter. Empirically, we find that this optimistic distributional targeting strategy effectively guides exploration; while it does not always lead to trajectories that meet the optimistic y^+, it consistently outperforms fixed target y^* and gradually shifts the distribution of collected returns toward higher values, as shown in Fig. 2.

The online finetuning process is structured in stages. In each stage, we first collect a set of new trajectories using the exploration policy described above. These newly collected trajectory-return pairs are then added to the replay buffer, enriching the dataset with recent, high-quality experiences. Following data collection, we fine-tune the GAC model on the updated replay buffer for a set number of steps, using the same ELBO maximization objective as in the pre-training phase. This cycle of

Table 1: **Offline results with and without Step-wise Rewards.** Comparison between the average normalized returns of our GAC method against several baselines, evaluated without any online finetuning. We assess the pure exploitation capability of our model, denoted as GAC-$\mathbb{E}[y]$, by optimizing over the entire latent space. For each of 5 independent seeds, we generate 100 evaluation trajectories from the pre-trained checkpoint and report the mean and standard deviation of their returns. The best result in each row is highlighted in bold.

Dataset	TD3+BC	IQL	CQL	DT	QDT	LPT	GAC-$\mathbb{E}[y]$
Results without step-wise rewards							
hopper-medium	27.6	35.1	23.3	57.3	50.7	58.5	**67.1**$\pm$0.8
hopper-medium-replay	10.6	13.9	7.7	50.8	38.7	71.2	**81.4**$\pm$1.1
walker2d-medium	67.8	49.1	0.0	69.9	63.7	77.8	**79.3**$\pm$0.9
walker2d-medium-replay	8.7	5.3	3.2	51.6	29.6	72.3	**78.9**$\pm$1.1
halfcheetah-medium	20.8	8.5	1.0	42.4	42.4	43.1	**43.3**$\pm$0.6
halfcheetah-medium-replay	12.8	5.2	7.8	33.3	32.8	39.6	**40.1**$\pm$0.4
Results with step-wise rewards							
hopper-medium	60.4	63.8	58.0	60.3	66.5	-	-
hopper-medium-replay	64.4	92.1	48.6	63.7	52.1	-	-
walker2d-medium	82.7	79.9	79.2	73.3	67.1	-	-
walker2d-medium-replay	85.6	73.7	74.1	60.2	58.2	-	-
halfcheetah-medium	48.1	47.4	44.4	42.1	42.3	-	-
halfcheetah-medium-replay	44.8	44.1	46.2	34.1	35.6	-	-

exploration, replay buffer update, and model fine-tuning is repeated, allowing GAC to progressively adapt and improve its performance through online interaction.

4 Experiments

We evaluate GAC on standard offline and offline-to-online reinforcement learning benchmarks from the D4RL Gym-MuJoCo suite (Halfcheetah, Hopper, and Walker2D). Our experiments are conducted in a setting where only total trajectory returns are available. We compare GAC against state-of-the-art methods and analyze the effectiveness of its different inference strategies for exploitation and exploration.

We compare GAC against strong baselines across several paradigms: offline RL (IQL[26], CQL[32], TD3+BC[24]), sequence modeling approaches (DT[7], ODT[8], QDT[43], LPT[12]), and classic online RL (PPO[44], SAC[3]) for reference. Notably, except for LPT, these methods were designed for step-wise rewards. We adapt them to our trajectory-return-only setting for a fair comparison, highlighting the distinct advantage of models that do not rely on dense reward signals.

Offline Pre-training. We report the offline pre-training results in Table 1, evaluated using the exploitative inference from Eq. (2) (termed GAC-$\mathbb{E}[y]$). Our model consistently outperforms all baselines across the tested environments. GAC's advantage stems from its unique generative approach. Unlike conventional methods rooted in temporal difference learning, GAC forgoes explicit per-step credit assignment. Instead, it learns an implicit understanding of action-outcome relationships via cross-attention across the latent token sequence. This allows the model to learn a smooth and structured latent space of behaviors, enabling it to compose novel, high-quality trajectories by interpolating between successful sub-sequences found in the training data. The robustness of this approach is particularly evident in Mujoco environments. While TD-based methods like IQL, CQL, and TD3+BC are heavily dependent on dense, step-wise rewards, GAC—much like other generative models such as DT, QDT, and LPT—maintains high performance even with only trajectory-level returns. This highlights a key strength of our framework: GAC achieves superior performance despite operating without the granular, step-wise reward signals that most of these powerful baselines rely on for effective training.

Inference Strategies. We analyze the practical implications of the different inference strategies introduced in Section 3.1. We compare three approaches: GAC-$\mathbb{E}[y]$ for pure exploitation via latent space optimization; GAC-y^*, which conditions on a standard high-value target return similar to DT[7]; and GAC-$p(y^+)$, our proposed exploration strategy that conditions on dynamically adjusted

Table 2: **Comparing inference objectives for pretrained GAC.** We report the mean and standard deviation of returns over 100 rollouts for exploitation (GAC-$\mathbb{E}[y]$), exploration (GAC-$p(y^+)$), fixed target conditioning (GAC-y^*), and sampling from the prior ($p(y|z)p(z)$).

Dataset	GAC-$\mathbb{E}[y]$	GAC-$p(y^+)$	GAC-y^*	$p(y\|z)p(z)$
hopper-medium	**67.6**±5.2	63.1±**8.3**	60.6±8.0	55.8±5.2
hopper-medium-replay	**81.8**±9.9	66.1±19.3	36.3±**23.8**	36.5±23.4
walker2d-medium	**80.2**±4.6	78.9±**8.9**	78.5±6.7	75.6±10.1
walker2d-medium-replay	**79.3**±10.9	63.1±**18.6**	59.5±14.1	48.9±26.6
halfcheetah-medium	**42.6**±4.2	41.1±**6.0**	40.3±4.2	41.2±7.5
halfcheetah-medium-replay	**39.8**±7.8	38.8±**8.9**	36.7±8.2	33.0±12.0

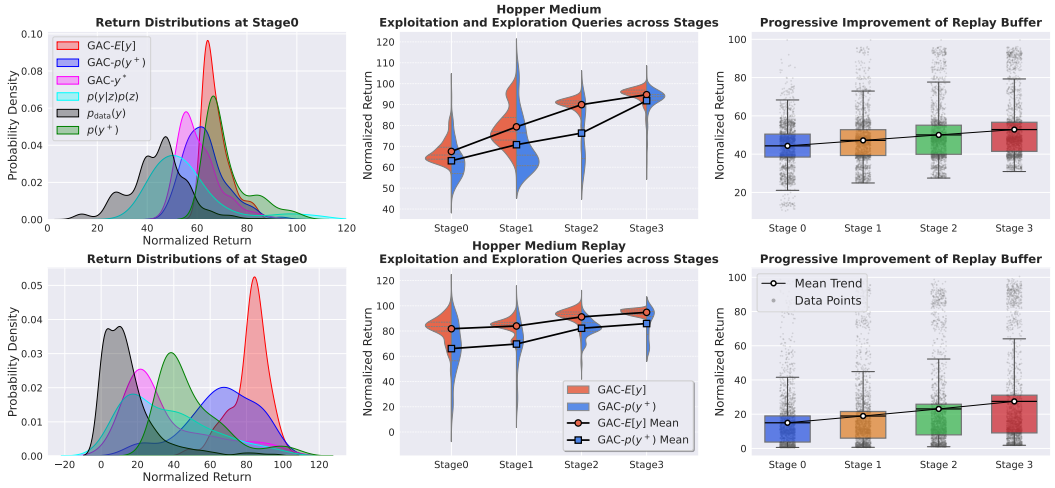

Figure 2: **Illustration of GAC's return distributions.** The left column displays the return distributions achieved by various inference methods after offline pre-training, highlighting the explorative and exploitative behavior. The middle column presents a split violinplot for GAC-$\mathbb{E}[y]$ and GAC-$p(y^+)$ inference, showing the performance gains of checkpoints and the explorative and exploitative behavior at three sequential fine-tuning stages. The right column visualizes the progressive improvement in the complete dataset's quality, as evidenced by the upward shift of the dataset's return distribution over the fine-tuning stages.

targets sampled from the replay buffer. This comparison is designed to evaluate the trade-offs between pure optimization, fixed-target conditioning, and our more adaptive, data-driven exploration method.

The return distributions collected by actors from different inference objectives, visualized in the left column of Fig. 2 and summarized in Table 2, highlight their distinct behaviors. The return distribution $p(y|z)$, generated from random prior latent plans from $p(z)$, is dispersed by principle and effectively covers the returns seen in the training data, confirming that the model has learned a well-structured latent space without collapsing to a single mode. In sharp contrast, the GAC-$\mathbb{E}[y]$ strategy produces a focused, low-variance distribution concentrated on high-certainty, high-return outcomes, underscoring its efficacy as a pure exploitation method. To encourage exploration, we sample target returns from $p(y^+)$, which apparently have better coverage at high-return and even out-of-distribution regions than GAC-$\mathbb{E}[y]$. Based on these targets, GAC-$p(y^+)$ achieves a high mean return but with significantly greater variance than GAC-$\mathbb{E}[y]$. While GAC-$p(y^+)$'s optimistic conditioning on y^+ leads to a delibrate mismatch between its resulting distribution and $p(y^+)$, it reflects a realistic improvement over the $p_{\text{data}}(y)$, showcasing successful guided exploration toward better outcomes. As GAC-$p(y^+)$'s target distribution is anchored at the data distribution, it also outperforms GAC-y^*, whose reliance on a single fixed target often leads to overestimation (*optimism bias*) of the outcomes.

Online Fine-tuning. Table 3 reports the final returns compared to other methods in most environments after fine-tuning with just 300 online trajectories. While the gains are relatively small in some tasks, this is often a consequence of its higher initial offline performance. Notably, GAC remains highly effective even without dense, step-wise rewards—a setting where pure online RL methods like PPO and SAC typically fail. Unlike methods that rely on explicit optimism or conservatism, GAC

Table 3: **Online training results with and without step-wise rewards.** Comparison of normalized returns before and after online fine-tuning. We fine-tune GAC for 3 stages with data collected by GAC-$p(y^+)$ and report the final GAC-$\mathbb{E}[y]$ performance from five seeds. The best final result for each dataset is highlighted in bold.

Dataset	Online RL		ODT		CQL		IQL		LPT		GAC	
	PPO	SAC	online	δ	online	δ	online	δ	online	δ	online	δ
Results without step-wise rewards												
hopper-m	11.5	9.8	57.6	0.3	29.6	6.3	25.0	-10.1	64.8	6.3	**94.1**$\pm$1.1	27.0
hopper-m-r	-	-	65.2	14.4	8.4	0.7	12.6	-1.3	72.4	1.2	**95.5**$\pm$0.8	14.1
walker2d-m	9.1	4.0	70.7	0.8	1.9	1.9	50.1	1.0	79.5	1.7	**84.7**$\pm$0.3	5.4
walker2d-m-r	-	-	57.3	5.7	0.5	-2.7	6.9	1.6	79.0	6.7	**84.6**$\pm$0.5	5.7
halfcheetah-m	1.2	1.9	40.7	-1.7	2.8	1.8	8.9	0.4	43.2	0.1	**44.3**$\pm$0.1	1.0
halfcheetah-m-r	-	-	24.4	-8.4	6.4	-1.4	7.4	2.2	40.6	1.0	**40.8**$\pm$0.1	0.7
Results with step-wise rewards												
hopper-m	77.8	75.0	97.5	30.6	60.4	2.4	66.8	3.0	-	-	-	-
hopper-m-r	-	-	88.9	2.3	56.3	7.7	96.2	4.1	-	-	-	-
walker2d-m	43.9	47.5	76.8	4.6	79.6	0.4	80.3	0.4	-	-	-	-
walker2d-m-r	-	-	76.9	4.0	75.8	1.7	70.6	-3.1	-	-	-	-
halfcheetah-m	25.0	24.4	42.4	-0.6	45.6	1.2	47.4	0.0	-	-	-	-
halfcheetah-m-r	-	-	40.2	0.4	43.0	-3.2	44.1	0.0	-	-	-	-

improves by leveraging its principled exploration strategy. This strategy systematically probes for trajectories with higher returns by conditioning on targets set slightly beyond the empirical best of the current dataset. When these exploratory rollouts yield higher-performing trajectories, they are added to the replay buffer. This process gradually shifts the collected data distribution toward superior outcomes, which is exemplified in the Hopper task where our method achieves its most significant performance gain. To visualize this dynamic process, Fig. 2 plots the return distributions from different inference objectives across several fine-tuning stages. The figure clearly illustrates the interplay between exploration and exploitation, tracking the progressive performance improvement via the continuous, positive shift in the collected data's return distribution. The implementation details and more results can be found in Appendix A.

5 Conclusion

In this paper, we introduced Generative Actor-Critic (GAC), a novel framework that decouples decision-making from generative decision-modeling by learning the joint distribution of trajectories and their returns, $p(\tau, y)$. Our latent variable-based instantiation enables principled inference strategies for both exploitation and exploration, leading to state-of-the-art offline performance and highly efficient online adaptation on Gym-MuJoCo benchmarks in the absence of step-wise rewards. Ultimately, GAC provides an effective framework that bridges generative modeling and reinforcement learning, opening a promising path toward building more capable and adaptable autonomous agents.

References

[1] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[2] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[4] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[5] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

[6] Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, and Remi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. In *International Conference on Learning Representations*, 2018.

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling, 2 june 2021. *URL http://arxiv. org/abs/2106.01345*, 2021.

[8] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

[9] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[10] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

[11] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

[12] Deqian Kong, Dehong Xu, Minglu Zhao, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, Sirui Xie, and Ying Nian Wu. Latent plan transformer for trajectory abstraction: Planning as latent space inference. *Advances in Neural Information Processing Systems*, 37: 123379–123401, 2024.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[14] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[16] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[17] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *32nd Conference on Uncertainty in Artificial Intelligence 2016, UAI 2016*, pages 202–211. Association For Uncertainty in Artificial Intelligence (AUAI), 2016.

[18] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[19] Andrew Kachites McCallum. *Reinforcement learning with selective perception and hidden state*. University of Rochester, 1996.

[20] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

[21] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

[22] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[23] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

[24] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[25] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

[26] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[27] Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, and Ya-Qin Zhang. Proto: Iterative policy regularized offline-to-online reinforcement learning. *arXiv preprint arXiv:2305.15669*, 2023.

[28] Alex Beeson and Giovanni Montana. Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning. *arXiv preprint arXiv:2211.11802*, 2022.

[29] Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

[30] Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

[31] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[32] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33: 1179–1191, 2020.

[33] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in cognitive sciences*, 16 (10):485–488, 2012.

[34] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[35] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[36] Aoyang Qin, Feng Gao, Qing Li, Song-Chun Zhu, and Sirui Xie. Learning non-markovian decision-making from state-only sequences. *Advances in Neural Information Processing Systems*, 36:6596–6618, 2023.

[37] Kevin P Murphy. Machine learning: A probabilistic perspective (adaptive computation and machine learning series). *The MIT Press: London, UK*, 2018.

[38] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *the Journal of machine Learning research*, 14(1):1303–1347, 2013.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[40] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[41] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

[42] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[43] Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pages 38989–39007. PMLR, 2023.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[45] Ying Li, Lei Cheng, Feng Yin, Michael Minyi Zhang, and Sergios Theodoridis. Overcoming posterior collapse in variational autoencoders via em-type training. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[46] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Table 4: GAC Hyperparameters.

Architecture-Related	Range
Embedding dimension	choice(64, 128, 192, 256, 512)
Number of latent tokens	choice(1, 2, 4, 8)
Number of attention heads	choice(1, 2, 4, 8)
Number of decoder layers	choice(1, 2, 3, 4)
Number of encoder layers	choice(1, 2, 3, 4)
Context length	choice(1, 4, 16, 32, 64)
Architecture-Unrelated	**Range**
Outer-loop learning rate	interval(1e-4, 1e-2)
Outer-loop weight decay	interval(1e-4, 1e-2)
Outer-loop training steps	choice(1, 5, 10, 20)
Inner-loop learning rate	interval(0.01, 0.5)
Batch size	choice(800, 700, 600, 500)

Table 5: The best GAC Architecture-Related Hyperparameters.

Parameter	h-m	h-m-r	w-m	w-m-r	ha-m	ha-m-r
Embedding dimension	192	128	192	192	128	256
Number of latent tokens	4	8	4	4	8	4
Number of attention heads	4	4	2	8	1	4
Number of decoder layers	3	4	3	4	3	3
Number of encoder layers	4	1	3	3	2	2
Context length	4	4	1	1	4	1

A More Details on MuJoCo

Inspired by Li et al. [45], we employ a more sophisticated training methodology than that of LPT [12]. This approach aims to tighten the approximation between the Evidence Lower Bound (ELBO) and the log-likelihood, and mitigating the issue of posterior collapse. Specifically, for each data point, we conduct inner-loop optimization until convergence, which is defined by a maximum of 100 training steps and an early-stopping threshold of 1e-4. In the outer loop, unlike the single-step approach in [12], we treat the number of training steps as a tunable hyperparameter. Given that GAC decouples decision-modeling from decision-making, we can directly assess the quality of our generative model by its loss value, rather than relying on immediate policy evaluation in the environment. Consequently, we leverage the Optuna framework [46] for hyperparameter optimization, with the objective of maximizing the ELBO. The complete set of hyperparameters subject to tuning is detailed in Table 4. Following the pre-training phase, architecture-related hyperparameters are fixed, while the remaining parameters are optimized during each subsequent fine-tuning stage. Using the optimal hyperparameters identified (see Table 5 and Table 6[1]), we then configure the inner-loop inference with more stringent convergence criteria: a maximum of 1000 steps and an early-stopping threshold of 1e-5.

For GAC-$p(y^+)$ to collect online data, we introduce another two parameters to obtain a target return set. We perform a data processing procedure on the current return dataset, which we denote as D. This process involves a quantile-based filtering and a subsequent transformation. The procedure is parameterized by two tunable hyperparameters: the quantile threshold, q, and an additive constant, Δy. First, we determine the q-th quantile of the dataset D, which we denote as y_q. All data points in D that are greater than y_q are selected to form a new subset, $D_{\text{sub}} = \{y \in D \mid y > y_q\}$. We finally randomly sample from D_{sub} and plus Δy to obtain the dynamic target return as $\{y + \Delta y \mid y \in D_{\text{sub}}\}$. This procedure allows us to isolate and transform the upper tail of the data distribution, with the specific threshold and transformation intensity controlled by the hyperparameters q and Δy, respectively. We present these additional parameters for inference and the standard target return for each environment

[1]The hyperparameters with interval range are fp64 after searching, only displayed four decimal places in Table 6. 'h', 'w', 'ha', 'm' and 'r' denotes 'hopper', 'walker2d', 'halfcheetah', 'medium' and 'replay' respectively.

Table 6: The best GAC Architecture-Unrelated Hyperparameters.

Parameter	h-m	h-m-r	w-m	w-m-r	ha-m	ha-m-r
Stage 0 (pre-training)						
Outer-loop learning rate	0.0010	0.0003	0.0023	0.0026	0.0026	0.0012
Outer-loop weight decay	0.0002	0.0056	0.0027	0.0011	0.0020	0.0071
Outer-loop training steps	10	1	5	1	10	10
Inner-loop learning rate	0.2365	0.2058	0.3980	0.0795	0.0501	0.0452
Batch size	700	500	600	500	500	700
Stage 1 (fine-tuning)						
Outer-loop learning rate	0.0002	0.0018	0.0008	0.0017	0.0062	0.0009
Outer-loop weight decay	0.0002	0.0026	0.0029	0.0062	0.0044	0.0060
Outer-loop training steps	1	20	1	20	5	20
Inner-loop learning rate	0.0458	0.2116	0.1223	0.0712	0.2363	0.2836
Batch size	800	700	800	500	700	500
Stage 2 (fine-tuning)						
Outer-loop learning rate	0.0004	0.0010	0.0013	0.0014	0.0093	0.0013
Outer-loop weight decay	0.0006	0.0007	0.0022	0.0050	0.0001	0.0002
Outer-loop training steps	1	1	5	20	5	5
Inner-loop learning rate	0.1018	0.0712	0.0124	0.1691	0.1435	0.2154
Batch size	700	500	800	500	600	700
Stage 3 (fine-tuning)						
Outer-loop learning rate	0.3755	0.0005	0.0007	0.0010	0.0039	0.0022
Outer-loop weight decay	0.0077	0.0001	0.0027	0.0007	0.0023	0.0015
Outer-loop training steps	5	5	20	1	20	20
Inner-loop learning rate	0.0023	0.010	0.0337	0.0712	0.1193	0.1984
Batch size	600	500	600	500	500	600

Table 7: The Inference Hyperparameters.

Parameter	h-m	h-m-r	w-m	w-m-r	ha-m	ha-m-r
Standard target return	3600	3600	6000	6000	5000	5000
Stage0 $(q, \Delta y)$	(0.8,10)	(0.8,10)	(0.8,10)	(0.8,10)	(0.6,3)	(0.6,3)
Stage1 $(q, \Delta y)$	(0.8,10)	(0.8,10)	(0.8,10)	(0.8,10)	(0.6,3)	(0.6,3)
Stage2 $(q, \Delta y)$	(0.8,10)	(0.8,10)	(0.6,3)	(0.6,5)	(0.6,3)	(0.6,1)
Stage3 $(q, \Delta y)$	(0.8,10)	(0.8,10)	(0.6,3)	(0.6,5)	(0.6,3)	(0.6,1)

in Table 7. We finally present all of the inference results for all Gym-MuJoCo tasks with GAC-$\mathbb{E}[y]$, GAC-$p(y^+)$, GAC-y^* and PRIOR inference strategies for one seed in Fig. 1.

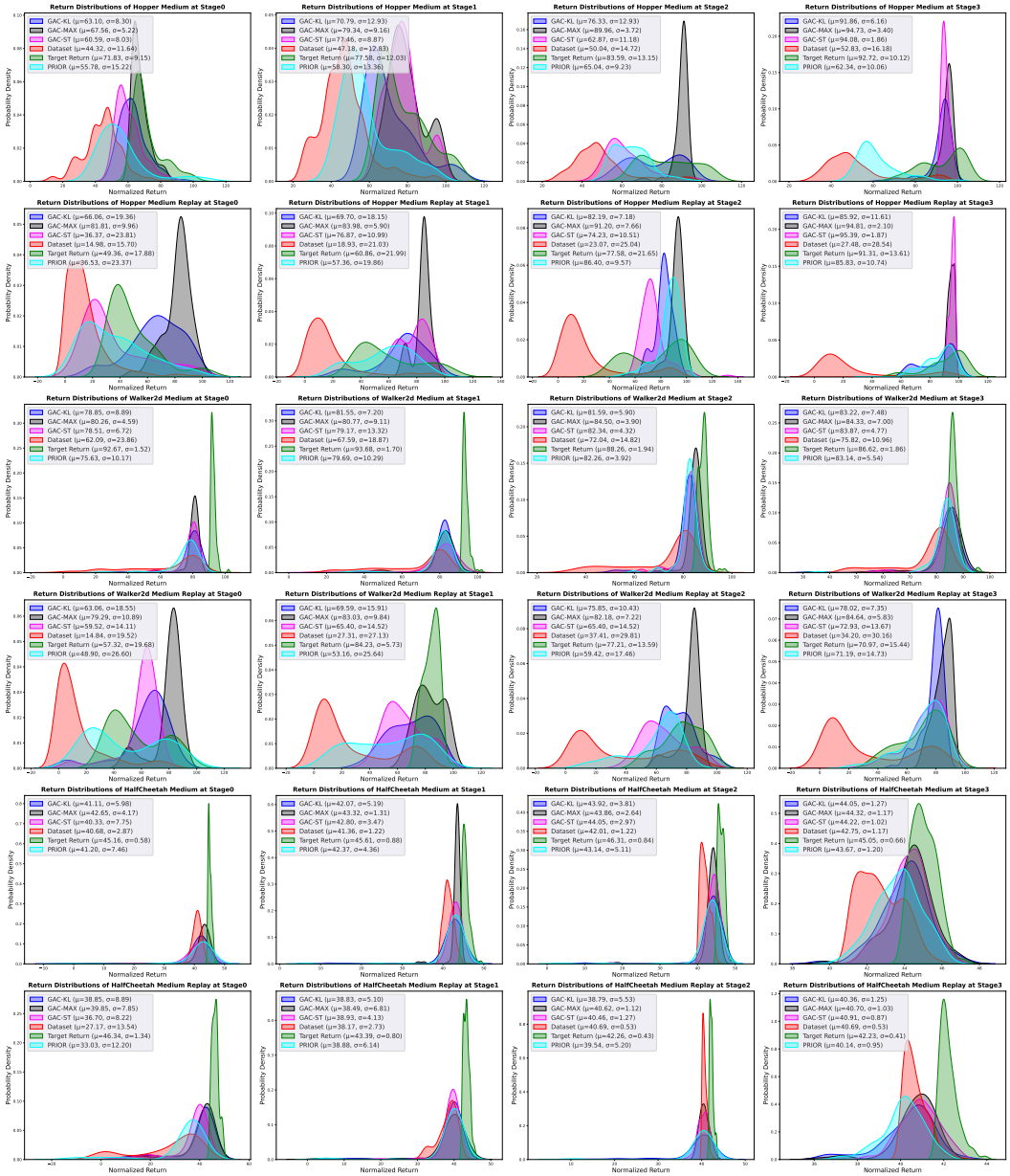

Figure 1: Complete MuJoCo Results.