Online Planning with Offline Pretrained All-in-One World Model

Anonymous Authors

***Abstract*—Recent work in Offline Reinforcement Learning (RL) has shown that an all-in-one world model pretrained offline via a masked auto-encoding objective can effectively capture the relationships between different modalities (e.g., states, actions, rewards) within trajectory datasets. However, this model's full potential has not been exploited during deployment, where the agent must generate an optimal policy rather than merely reconstruct masked tokens. Since the pretrained model subsumes both a Policy Model and a World Model under appropriate mask patterns, we propose leveraging it for *online planning* via Model Predictive Control (MPC) at test time, using the model's own predictive capability to guide action selection. Empirical results on D4RL and RoboMimic show that our online planning framework significantly improves decision-making performance of the pretrained model without any additional parameter training. Furthermore, the framework extends naturally to Offline-to-Online (O2O) RL and Goal-Reaching RL, yielding more substantial gains when an online interaction budget is available and better generalization when diverse task targets are specified.**

***Index Terms*—offline reinforcement learning, all-in-one world model, online planning, model predictive control, goal reaching**

I. Introduction

The Masked Modeling paradigm has a simple, self-supervised training objective: predicting a randomly masked subset of the original sequence. It has become a powerful technique for generation or representation learning for sequential data, e.g., language tokens [1] or image patches [2]. Unlike autoregressive models like GPT [3], which condition only on the past context in the "left", bidirectional models trained with this objective learn to model the context from both sides, leading to richer representations and deeper understandings of the data's underlying dependencies.

Given that a sequential decision-making trajectory inherently involves a sequence of states s and actions a, and other optional augmented properties like return-to-go (RTG) g [4] or approximate state-action value v [5] across T timesteps, the mask modeling paradigm can be adapted easily for sequential decision-making tasks. For example, in the case of Reinforcement Learning, the policy output $\mathbb{P}(a|s)$ at each time step can be regarded as predicting a masked action a conditioned on given states s. Moreover, recent works [6]–[8] have demonstrated that a unified bidirectional trajectory model (BTM) pretrained with a highly random masking pattern can be applied zero-shot in various downstream tasks. Different reconstruction tasks can be deliberately created by applying appropriate masks to different modalities—whether states, actions, or rewards—during inference time.

However, the inherent flexibility and versatility of models trained with random masking techniques have not been fully exploited in deployment settings. Previous research has highlighted the multitasking capabilities of Bidirectional Trajectory Models (BTMs) by assigning **one** single specific mask pattern to individual tasks, such as the RCBC mask commonly used in offline RL after the pretraining phase. Our findings, in contrast, suggest that integrating **multiple** capabilities such as short-term reward and long-term return prediction, along with forward dynamics, could significantly enhance decision-making. These capabilities allow the agent to explicitly evaluate action candidates and determine the optimal one, rather than merely relying on implicit mappings from expected returns to policies.

Building on these insights, we introduce the **M^{3}PC** framework: Enhancing Decision-Making by using the **M**asked **M**odel itself for test-time **M**odel **P**redictive **C**ontrol. Our framework decomposes decision-making tasks into a series of simpler steps in a typical sample-based MPC style: sampling potential actions, inferring possible future states, evaluating these actions based on predicted outcomes, and selecting the final optimal action. We then demonstrate how a pretrained model, equipped with our adaptation and ensemble of masks, can efficiently and effectively handle these subtasks. Our empirical results demonstrate that, by using M^{3}PC for final decision-making, the same pretrained model can get substantial decision quality improvement in offline RL and goal-reaching RL, outperforming traditional single-mask models. Furthermore, M^{3}PC supports sample-efficient online finetuning—a capability rarely seen in previous sequential modeling agents. By fully leveraging the potential of a pretrained BTM, M^{3}PC evolves the model from a multitasking framework into an inference-phase self-enhancing, and a finetuning-phase self-improving generalist agent. We summarize our results in Figure 1 and highlight our contributions as:

- We present M^{3}PC, a novel framework that utilizes mask ensembles to address complex decision-making tasks, effectively leveraging the multitasking abilities of a pretrained bidirectional trajectory model (BTM).
- We demonstrate that M^{3}PC not only improves the test-time performance of the same pretrained BTM in offline RL by 6.0%, but also enables efficient finetuning through online interactions with environments, outperforming specialized offline-to-online (O2O) RL algorithms, such as ODT, by 26.0%.
- We show that M^{3}PC can be adapted for goal-reaching

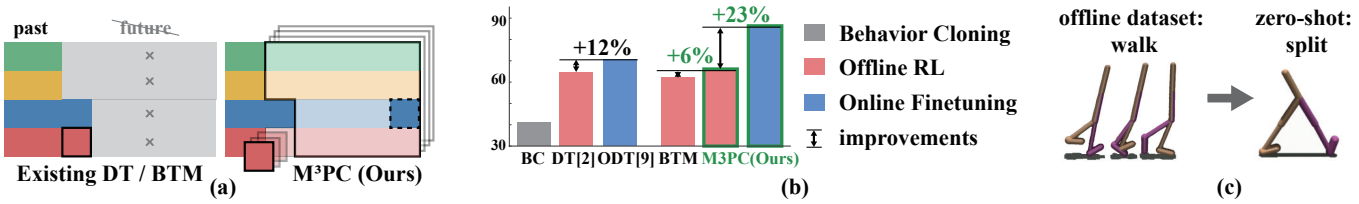

Fig. 1: **Benefits of equipping pretrained bidirectional trajectory model with our test-time M^{3}PC. (a)** Instead of generating actions solely based on history context, we leverage the full capacity of the masked pretrained model to predict future outcomes (e.g. states, rewards, returns) as a test-time self-enhanced decision making approach. Such a MPC framework can be used to achieve higher return at inference time or to reach a given goal state (in dashed square block) even unseen during offline training. **(b)** Forward M^{3}PC achieves better offline learning performances, using the same model without any finetuning, and gains better O2O improvement when online finetuning is allowed after offline pretrain. **(c)** Backward M^{3}PC unlocks zero-shot goal reaching capability. Given a desired state, the walker agent can split its legs to a large degree without any prior experience.

tasks, effectively guiding agents to specified goal states—even when these states are out-of-distribution relative to the datasets used for pretraining.

II. Related Work

Transformers for Sequential Decision Making. The Transformer [9] architecture has been extensively applied in sequential decision-making tasks such as reinforcement learning (RL) [4], [10], [11] and imitation learning (IL) [12]–[16]. Representative work such as Decision Transformer (DT) [4] and its variants [5], [17] learn a return-conditioned policy using a causal-masked Transformer. Recent studies [6]–[8] utilize a bidirectional Transformer to model trajectories, highlighting the model's versatility enhanced by the mask prediction training objective. These studies focus on the potential of trajectory Transformers to unify various decision-making tasks, typically employing a unique mask pattern tailored to each specific downstream task. Building upon these insights, our work proposes harnessing the functional versatility of pretrained models to enhance decision-making. More specifically, we investigate whether utilizing two or more mask patterns can lead to improved decision-making within a single downstream task.

Offline RL with Online Finetuning. Traditional off-policy RL algorithms often suffer from bootstrapping error accumulation [18], [19]. To mitigate these issues, most offline RL algorithms employ regularization techniques to mitigate errors caused by out-of-distribution actions [18]–[23]. However, finetuning an offline RL algorithm can be challenging due to its inherent conservatism and the offline-to-online data distribution shift [19], [24]. Many techniques such as value calibration [25], balanced replay [26] and policy expansion [27] have been investigated to improve the online sample efficiency. In parallel, some work [4], [17] following supervised learning (SL) paradigm can naturally ensure in-distribution learning but also suffer from poor online sample efficiency [28]. Our approach adheres to the SL paradigm while incorporating a DP-based module to improve online sample efficiency.

Model-based RL. Learning a dynamics model of the environment can be used for policy learning [29]–[31] or planning [32]–[35]. Recent work has explored the feasibility of MPC in online RL [36]–[41]. Similar planning methods have also been tailored for offline RL using techniques such as behavior cloning regularization [42] and trajectory pruning [43], [44]. Instead of maintaining separate world and policy models, Trajectory Transformer (TT) [10] frames RL as a sequential modeling problem and performs beam search planning based on return heuristics. Our work follows a similar paradigm but leverages a bidirectional Transformer and mask autoencoding to enable a more flexible and computationally efficient planning process.

III. Preliminary

We consider the environment as a Markov Decision Process (MDP), formally defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \rho_0 \rangle$. In this notation, $\mathcal{S}$ represents the state space, and $\mathcal{A}$ represents the action space. The transition probability distribution, $P(s_{t+1} \mid s_t, a_t)$, defines the likelihood of transitioning from state s_t to state s_{t+1} given action a_t. The reward function, $R(s_t, a_t)$, assigns a reward for each action taken in a particular state. The discount factor, denoted by γ, quantifies the preference for immediate rewards over future rewards. The maximum episode length, also referred to as the horizon of the MDP, is denoted as H.

Additional notations are introduced to adapt RL to sequential modeling. We denote the training data distribution as $\mathcal{T}$, which may be dynamic when the agent interacts with the environment. A trajectory τ, consisting of T states, actions, RTGs and rewards is represented as $\tau = (s_1, g_1, a_1, r_1, \cdots, s_T, g_T, a_T, r_T)$. Note that some other properties can also be directly or indirectly accessed from the training data such as next-states $(s'_1, \cdots, s'_T)$, estimated values $(v_1, \cdots, v_T)$ for state-action pairs, but we do not model these modalities on the Transformer.

IV. Method

This section details how we leverage a bidirectional trajectory model's versatile prediction capabilities within the M^{3}PC framework to enhance an agent's decision-making. In a typical MPC process, the system repeatedly solves an optimization problem to identify the best sequence of actions

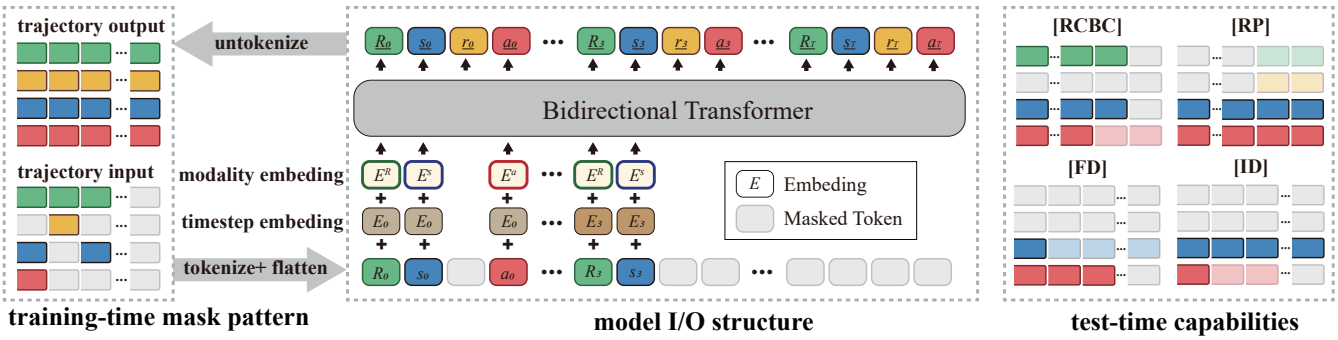

Fig. 2: **Model overview.** The bidirectional trajectory model is pre-trained using MAE loss that aims to reconstruct the whole MDP trajectory given a **[Random]** masked trajectory. After pretraining, the model shows multiple capabilities by applying different test-time masks. E.g., **Return-Conditioned Behaviour Clone [RCBC] Mask:** Predict actions given states, expected return and context trajectory. **Reward and Return Prediction [RP] Mask:** Predict intermediate rewards and future return given states and actions. **Forward Dynamics [FD] Mask:** Predict future states given current state and future actions. **Inverse Dynamics [ID] Mask:** Infer actions needed to perform a given state path. For MDP-elements not related to the given task (e.g., rewards during [RCBC]), we omit and mark them as gray.

over a finite horizon by evaluating the outcomes of these actions and then executing the action at the current timestep. The following subsections describe our approach to adapting a BTM to perform these MPC steps: First, we enable the BTM to **reconstruct actions with uncertainty**, allowing us to sample from a distribution of action proposals. Next, we demonstrate how to use different masking patterns for **forward** or **backward** prediction for MDP sequence elements. These predictions serve as references for evaluating the expected outcomes of action proposals, which we use to determine the optimal action to execute. As illustrated in Figure 3, by breaking down decision-making into these structured steps and using the BTM for versatile predictions, our M^{3}PC framework enhances the agent's ability beyond simply imitating behaviors observed in offline data, e.g., achieving higher reward incomes or diverse goals which typically fall in offline RL and goal reaching domains, respectively.

Bidirectional Trajectory Model. We illustrate the model architecture and how it processes a masked MDP trajectory in Figure 2. To perform masked trajectory modeling, we first flatten and tokenize the different elements of the raw trajectory sequence. This tokenization involves three components: a modality-specific encoder that lifts elements from the raw modality space to a common representation space, along with the addition of timestep embeddings and modality-type embeddings. These components collectively enable the Transformer to distinguish between different sequence elements.

We employ an encoder-decoder architecture with both the encoder and decoder being bidirectional Transformers. The tokenized and flattened trajectory is fed into the Transformer encoder, where only unmasked tokens are processed. The decoder then processes the full trajectory sequence, leveraging values from the encoder when available or substituting a mask token when not. The decoder is trained to reconstruct the original sequence, including the unmasked tokens.

Training-phase Mask Pattern. Inspired by previous work [8], [45], we employ a two-step masking pattern for training. Firstly, we randomly mask a proportion of elements in the trajectory τ. Secondly, we mask all elements to the right of a randomly chosen position. By learning to predict the masked elements, the model is trained to handle temporal dependencies and infer outcomes based solely on past events.

Uncertainty-Aware Action Reconstruction. To equip the agent with robust decision-making capabilities beyond mere imitation, our method employs uncertainty-aware action reconstruction rather than predicting the masked action deterministically. The primary focus of MAE lies in perfectly reconstructing each token of the sequence, typically by optimizing a Mean Squared Error (MSE) loss. This inherently leads to deterministic action reconstruction, which limits the agent's ability to account for uncertainties associated with the actions.

To address this limitation, we propose reconstructing an uncertainty-aware action distribution $\mathcal{A}$ by minimizing a Negative Log Likelihood (NLL) loss $J(\theta)$ denoted by

$$J(\theta) = \frac{1}{T}\mathbb{E}_{\tau\sim\mathcal{T}}\left[\sum_{t=1}^{T} -\log P_\theta(a_t|\texttt{Masked}(\tau))\right]. \quad (1)$$

Inspired by ODT [17], we additionally impose a lower bound on trajectory-level action entropy $\mathcal{H}_\theta^{\mathcal{T}}$ to encourage the agent's online exploratory behavior. The overall constraint problem is formally defined as

$$\min_\theta \; J(\theta) \quad \text{s.t.} \quad \mathcal{H}_\theta^{\mathcal{T}} \geq \beta,$$
$$\mathcal{H}_\theta^{\mathcal{T}} = \frac{1}{T}\mathbb{E}_{\tau\sim\mathcal{T}}\left[\sum_{t=1}^{T} H[P_\theta(a_t \mid \texttt{Masked}(\tau))]\right]. \quad (2)$$

where $H[\cdot]$ denotes the Shannon entropy of the distribution and β represents the predefined target entropy. To avoid explicitly handling the inequality constraint, we solve the Lagrangian dual problem of Equation 2

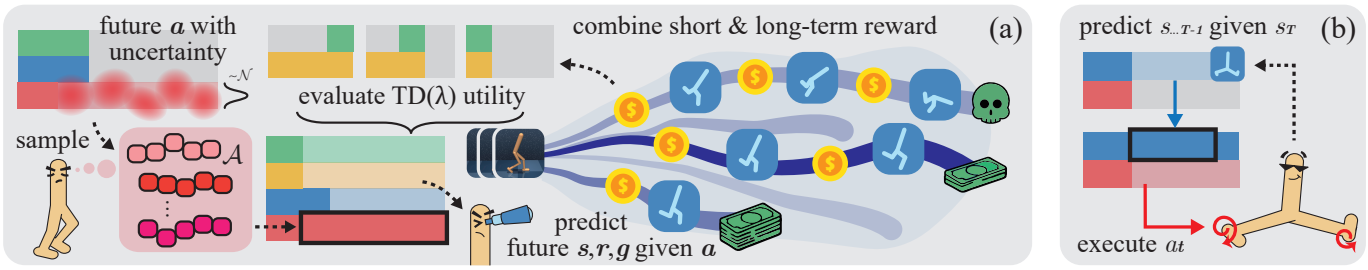

Fig. 3: **Leverage the Masked Model itself for test-time Model Predictive Control.** Our pipeline utilizes BTM's versatile inference capabilities to enhance decision making. **(a) Forward M^{3}PC.** We employ [RCBC], [FD] and [RP] masks to build an MPC pipeline for planning, prediction, and action resample. **(b) Backward M^{3}PC.** Given a goal state that we finally want to reach, we first use Path Inference [PI] mask to infer the waypoint-states, followed by an Inverse Dynamic [ID] mask to get the action sequence conditioned on those waypoints, and finally execute the first one.

Forward M^{3}PC for Reward Maximization. A bidirectional trajectory model agent has demonstrated zero-shot ability in offline RL tasks when equipped with an [RCBC] mask as shown in previous work [6], [8]. By predicting actions conditioned on states and RTGs, the agent generates actions by imitating trajectories with similar RTGs in offline data. The performance of this imitative behavior is inherently upper-bounded by the best trajectory in the offline data.

To address this limitation, we propose refining the decision-making process by implementing an explicit reward-maximization procedure using the forward dynamics function and the return and reward prediction functions provided by the unified trajectory model. Typically, this process divides decision-making into three substeps: generating action proposals, rolling out the future, and selecting action proposals based on their potential utilities. Suppose we have access to both intermediate and long-term reward estimation for candidate action sequence $a_{t:T}$, represented by $r_{t:T}$ and $g_{t:T}$, respectively. We define the TD(λ)-style utility U for this candidate action as follows

$$\begin{aligned} U &= (1-\lambda) \sum_{n=0}^{T-t-1} \lambda^n G_{t:t+n} + \lambda^{T-t} G_{t:T}, \\ G_{t:t+n} &= \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n g_{t+n}, \end{aligned} \tag{3}$$

where the decay parameter λ determines the weights of longer horizon estimates that contribute to the final result, helping trade off errors from dynamics predictions and value estimates. We construct a categorical distribution $\mathcal{P}$ using *softmax* for proposal selection:

$$P[i] = \frac{\exp(\xi U^i)}{\sum_j \exp(\xi U^j)}, \quad \forall i \in [1, \cdots, N], \tag{4}$$

where ξ denotes the *softmax* temperature. Notably, M^{3}PC requires only two prediction steps for planning at each timestep. Leveraging the bidirectional nature of Transformers and the masked autoencoding paradigm, M^{3}PC can predict all future actions given current states and all future states given future actions in parallel. This parallel prediction capability mitigates the computational cost's linear growth with respect to the planning horizon, which is commonly observed in planning algorithms such as beam search in TT [10] or CEM in TD-MPC [41]. We detail the decision-making process for reward-maximization. Since RTG value is a trajectory-wise Monte Carlo estimation, it becomes uninformative when datasets' behavior policies are diverse. We can optionally extend M^{3}PC by replacing RTG guidance with a transition-wise value for a better heuristic, calculated with a standalone value estimator updated via dynamic programming as in IQL [21].

Using the 'Utility' metric to estimate future actions before they were taken, forward M^{3}PC can also adapt to an exploration strategy in the subsequent online finetuning phase, where (3) is used again. During the offline-to-online process, instead of executing the expectation in categorical distribution (4), the M^{3}PC agent samples actions from the candidate set according to the possibilities proportional to their utility. This introduces stochasticity, maintaining overall superior actions while ensuring diversity in the experience collected during exploration, thereby balancing exploration and exploitation.

Backward M^{3}PC for Goal Reaching. The ability of a BTM to infer past tokens conditioned on future events sets it apart from GPT-based models. This feature is particularly advantageous for implementing MPC from a reverse or "backward" perspective when the objective is to achieve a specified goal state. Unlike the goal-reaching mask proposed in previous works [6], [7], which masks all elements along the trajectory except the current and final states to reconstruct the action at the current timestep, we leverage the BTM's bidirectional conditioning capability to inpaint a transition path that guides action selection. We refer to this method as backward M^{3}PC.

Specifically, the backward M^{3}PC approach uses a Path Inference (PI) mask (Figure 3(b)) to guide the model in predicting a sequence of intermediate states leading to the goal. Once a path is established, the model employs an Inverse Dynamics (ID) mask to deduce the necessary actions to transition between consecutive states along the predicted path. This approach eliminates the need to generate a large number

of candidates and roll out each one. Instead, it implicitly performs the same function as traditional MPC by selecting the first action in a sequence that most satisfies the given goal.

V. Experiments

Our experiments aim to answer the following questions:

Q1: Can forward M^{3}PC enable the (same) agent to achieve higher accumulated rewards in offline RL and subsequent online finetuning?

Q2: Can backward M^{3}PC enable the agent to perform diverse tasks given target states?

Q3: How does each algorithmic component contribute to M^{3}PC?

Q4: Is the pretrained model capable enough to perform M^{3}PC in more complex environments demanding interaction with external objects, e.g. manipulation?

Tasks and Datasets. To answer these questions, we utilize **D4RL** and **RoboMimic** dataset suites. We apply three D4RL locomotion domains (`Hopper`, `Walker2d`, `HalfCheetah`) with two dataset types for each task: `medium(m)` and `medium-replay(m-r)`, used to benchmark our proposed forward M^{3}PC in offline RL and O2O settings. The RoboMimic encompasses three manipulation tasks (`Can`, `Lift`, `Square`). We utilize three official datasets (`can-pair`, `square-mh`, `lift-mg`) and two customized datasets (`can-lim`, `can-real`) to evaluate M^{3}PC's potential real-world application, particularly in robotic manipulation tasks.

Q1: Offline RL. We present the offline results of M^{3}PC-M and M^{3}PC-Q in Table I. Baselines include: (1) BC: behavior cloning; (2) TD3+BC [46]: off-policy RL with behavior cloning regularization; (3) IQL [21]: model-free algorithm avoiding bootstrapping errors via implicit Q-functions; (4) DT [4]: return-conditioned sequence modeling; (5) TT [10]: sequence modeling with beam search planning; and (6) BTM: the same pretrained model as ours using only the [RCBC] mask. M^{3}PC consistently outperforms BTM across all datasets. M^{3}PC-Q achieves competitive results against specialized offline RL algorithms by a considerable margin.

Online Finetuning. Under the O2O setup, we compare against IQL and ODT [17], a specially designed O2O method for DT. In Table I, we report performance with a 200K online sample budget. Our method outperforms the other two in all environments except `hopper-medium`. After fine-tuning, our total performance score is 31% higher than IQL and 26% higher than ODT, with finetuning improvements 123% more substantial than ODT.

Q2: Goal Reaching. To assess whether our proposed method can effectively guide an agent to specified goal states, we evaluate it on three tasks: (a) Halfcheetah flipping, (b) Walker performing splits, and (c) Hopper wiggling. We provide a sequence of consecutive subgoals to stay within the model's planning horizon.

These tasks deviate from the reward mechanisms in offline data but can be extrapolated from offline trajectories. Our results in Figure 4 illustrate that backward M^{3}PC enables the agent to generalize to diverse tasks rather than merely imitating offline experiences.

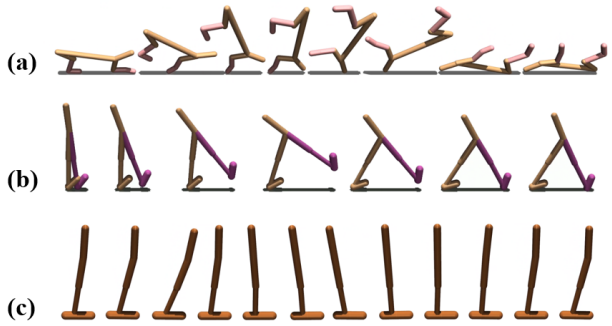

Fig. 4: **D4RL Goal Reaching.** (a) Halfcheetah flipping, (b) Walker splits, (c) Hopper wiggling. All behaviors are unseen during pretraining.

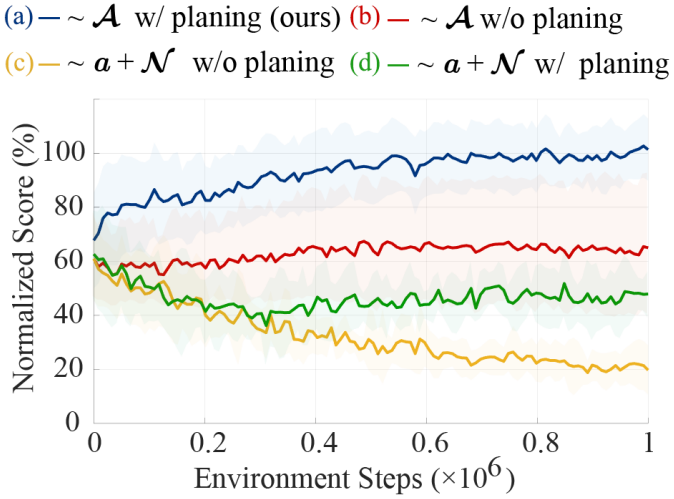

Fig. 5: **Ablation Study on Planning and Uncertainty-aware Action Reconstruction.** We ablate sample-based planning, uncertainty-aware action reconstruction, and both components for the online finetuning phase. Average over six datasets, five seeds. Shaded area: per-task std.

Additionally, we evaluated the BTM's goal-reaching ability using a single goal-reaching mask similar to [6], [7], which keeps only the current and goal states unmasked. However, this approach failed to consistently reach the goal state, underscoring the effectiveness of our model-based approach.

Q3: Ablation Studies.

We justify design choices in M^{3}PC's online finetuning phase by comparing: (a) M^{3}PC combining uncertainty-aware action reconstruction and planning-based resampling; (b) random sampling from $\mathcal{A}$ for exploration; (c) original BTM action reconstruction trained with MSE loss plus fixed noise $\mathcal{N}(\mathbf{0}, \sigma I)$ at the same entropy level; (d) planning-based resampling using (c)'s decisions. Results in Figure 5 highlight that uncertainty-aware policy is crucial for online stability, and forward planning significantly improves sample efficiency.

TABLE I: **D4RL Results: Offline and Online Finetuning.** Average normalized return without and with online finetuning (**200K** sample budget). M^{3}PC-M and M^{3}PC-Q use Monte Carlo return and Q-value guidance heuristics, respectively. Mean ± std. over 5 seeds. Best result per section in **bold**; greatest online improvement in green.

Dataset	Offline								Online Finetuning (200K)					
	BC	TD3+BC	IQL	DT	TT	BTM	M^{3}PC-M	M^{3}PC-Q	ODT			M^{3}PC-Q (Ours)		
									off.	*on.*	δ	*off.*	*on.*	δ
hopper-m	53.5	60.4	63.8	65.1	61.1	64.3	70.7 ±6.2	**73.6** ±5.6	67.0	**97.5**	+30.6	73.6	93.9	+20.3
walker2d-m	63.2	82.7	79.9	67.6	79.0	72.5	80.9 ±2.5	**86.4** ±2.6	72.2	76.8	+4.6	86.4	**91.9**	+5.5
halfcheetah-m	42.4	48.1	47.4	42.2	46.9	43.0	43.9 ±3.9	**51.2** ±0.7	42.7	42.2	−0.6	51.2	**69.3**	+18.1
hopper-m-r	29.8	64.4	**92.1**	81.8	91.5	75.3	80.4 ±5.2	78.3 ±16.2	86.6	88.9	+2.3	78.3	**103.5**	+25.2
walker2d-m-r	21.8	85.6	73.7	82.1	82.6	76.6	78.2 ±10.2	**92.2** ±2.4	68.9	76.9	+7.9	92.2	**105.2**	+13.0
halfcheetah-m-r	35.7	44.8	44.1	**48.3**	41.9	41.1	41.8 ±0.5	48.2 ±0.4	40.0	40.4	+0.4	48.2	**67.0**	+18.8
Total	246.4	386.0	401.0	387.1	403.0	372.8	395.9	**429.8**	377.4	422.7	+45.3	429.8	**530.8**	+101.0

TABLE II: **Offline Results on RoboMimic.** Task success rate of offline pretrained agents. Mean of 5 seeds (50 simulator / 20 real-world trials). BC and IQL excluded from real-world evaluation due to poor simulator performance.

Dataset	BC	IQL	DT	M^{3}PC
Can-Pair	0.64	0.34	0.94	**0.98** ±0.01
Square-MH	**0.53**	0.13	0.21	0.28 ±0.14
Lift-MG	0.65	0.29	**0.93**	0.77 ±0.07
Can-Lim	0.25	0.27	0.46	**0.54** ±0.16
Can-Real	—	—	0.50	**0.70** ±0.10

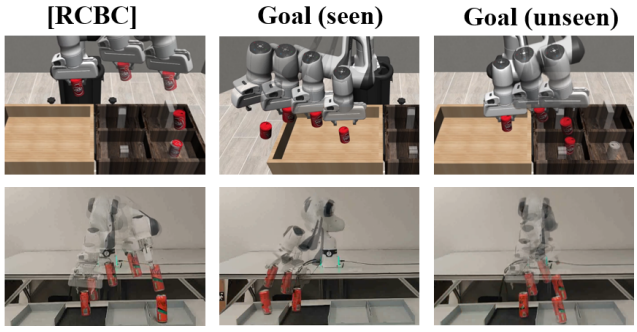

Fig. 6: **Skill Generalization in Can-Pick task.** Simulated environments (top) and real-world environments (bottom). Columns: original behavior (left), seen goal state (mid), unseen goal state (right).

Q4: Manipulation. In addition to self-body motion control tasks, we assess M^{3}PC on manipulation tasks requiring interaction with objects. We conducted experiments across three simulated RoboMimic tasks—Can, Square, and Lift—with varying complexity, using machine-generated (MG), mid-level human-demonstrated (MH), and paired positive-and-negative (Pair) demonstrations. We also created Can-Lim, a Can-Pick variant without gripper-to-can relative pose information, and tested on a real-world Can-Real task. Results are in Table II.

For generalization evaluation, we conducted a goal-conditioned experiment on Can-Picking (Paired dataset), containing 50% perfect demonstrations (can placed in right box) and 50% negative demonstrations (can thrown off table). As shown in Figure 6, specifying final goal states controls the agent's behavior. Furthermore, specifying a state numerically between two observed states generates actions enabling the agent to reach a previously unseen state—placing the can into an adjacent box.

VI. Discussions and Limitations

We propose M^{3}PC, a test-time MPC framework designed to enhance the inference performance of masked Transformers pretrained under offline RL settings. M^{3}PC offers: **(1) Improved Decision-Making without Further Training**: M^{3}PC improves decision-making at inference time with high computational efficiency. **(2) Enhanced Finetuning Efficiency:** With an additional online interaction budget, M^{3}PC achieves better final performance and outperforms the previous O2O approach ODT. **(3) Generalization Ability:** The framework generates actions that drive the agent toward unseen goal states in both simulated and real-world tasks.

Limitations for further investigation include: **(1) Pixel Observations:** Our framework is currently limited to state-based observations. Future work will explore pixel-based settings. **(2) Transformer Scalability:** It remains unclear whether increasing the Transformer's capacity would improve test-time planning.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[5] T. Yamagata, A. Khalil, and R. Santos-Rodriguez, "Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 989–39 007.

[6] M. Carroll, O. Paradise, J. Lin, R. Georgescu, M. Sun, D. Bignell, S. Milani, K. Hofmann, M. Hausknecht, A. Dragan *et al.*, "Uni [mask]: Unified inference in sequential decision problems," *Advances in neural information processing systems*, vol. 35, pp. 35 365–35 378, 2022.

[7] F. Liu, H. Liu, A. Grover, and P. Abbeel, "Masked autoencoding for scalable and generalizable decision making," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 608–12 618, 2022.

[8] P. Wu, A. Majumdar, K. Stone, Y. Lin, I. Mordatch, P. Abbeel, and A. Rajeswaran, "Masked trajectory models for prediction, representation, and control," in *International Conference on Machine Learning*. PMLR, 2023, pp. 37 607–37 623.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.

[11] K. Wang, H. Zhao, X. Luo, K. Ren, W. Zhang, and D. Li, "Bootstrapped transformer for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 748–34 761, 2022.

[12] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[13] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22 955–22 968, 2022.

[14] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[15] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 639–24 654, 2022.

[16] Z. Jia, V. Thumuluri, F. Liu, L. Chen, Z. Huang, and H. Su, "Chain-of-thought predictive control," *arXiv preprint arXiv:2304.00776*, 2023.

[17] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," in *international conference on machine learning*. PMLR, 2022, pp. 27 042–27 059.

[18] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[19] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[20] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[21] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[22] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified q-ensemble," *Advances in neural information processing systems*, vol. 34, pp. 7436–7447, 2021.

[23] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in neural information processing systems*, vol. 32, 2019.

[24] Z. Yu and X. Zhang, "Actor-critic alignment for offline-to-online reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 40 452–40 474.

[25] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[26] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[27] H. Zhang, W. Xu, and H. Yu, "Policy expansion for bridging offline-to-online reinforcement learning," *arXiv preprint arXiv:2302.00935*, 2023.

[28] D. Brandfonbrener, A. Bietti, J. Buckman, R. Laroche, and J. Bruna, "When does return-conditioned supervised learning work for offline reinforcement learning?" *Advances in Neural Information Processing Systems*, vol. 35, pp. 1542–1553, 2022.

[29] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," *arXiv preprint arXiv:1802.09081*, 2018.

[30] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," *Advances in neural information processing systems*, vol. 31, 2018.

[31] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[32] D. Silver, R. S. Sutton, and M. Müller, "Sample-based learning and search with permanent and transient memories," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 968–975.

[33] T. Walsh, S. Goschin, and M. Littman, "Integrating sample-based planning and model-based reinforcement learning," in *Proceedings of the aaai conference on artificial intelligence*, vol. 24, no. 1, 2010, pp. 612–617.

[34] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 7444–7453.

[35] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 129–14 142, 2020.

[36] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[37] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[38] Z. Wu, C. Yu, C. Chen, J. Hao, and H. H. Zhuo, "Plan to predict: Learning an uncertainty-foreseeing model for model-based reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 849–15 861, 2022.

[39] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[40] N. Hatch and B. Boots, "The value of planning for infinite-horizon model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7372–7378.

[41] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," *arXiv preprint arXiv:2203.04955*, 2022.

[42] A. Argenson and G. Dulac-Arnold, "Model-based offline planning," *arXiv preprint arXiv:2008.05556*, 2020.

[43] X. Zhan, X. Zhu, and H. Xu, "Model-based offline planning with trajectory pruning," *arXiv preprint arXiv:2105.07351*, 2021.

[44] M. Wang, R. Yang, X. Chen, H. Sun, M. Fang, and G. Montana, "Goplan: Goal-conditioned offline reinforcement learning by planning with learned models," *arXiv preprint arXiv:2310.20025*, 2023.

[45] Z. Zeng, C. Zhang, S. Wang, and C. Sun, "Goal-conditioned predictive coding for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[46] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.