Unsupervised Meta-Testing With Conditional Neural Processes for Hybrid Meta-Reinforcement Learning

Suzan Ece Ada [ID] and Emre Ugur [ID]

Abstract—We introduce Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP), a novel hybrid few-shot meta-reinforcement learning (meta-RL) method that uniquely combines, yet distinctly separates, parameterized policy gradient-based (PPG) and task inference-based few-shot meta-RL. Tailored for settings where the reward signal is missing during meta-testing, our method increases sample efficiency without requiring additional samples in meta-training. UMCNP leverages the efficiency and scalability of Conditional Neural Processes (CNPs) to reduce the number of online interactions required in meta-testing. During meta-training, samples previously collected through PPG meta-RL are efficiently reused for learning task inference in an offline manner. UMCNP infers the latent representation of the transition dynamics model from a single test task rollout with unknown parameters. This approach allows us to generate rollouts for self-adaptation by interacting with the learned dynamics model. We demonstrate our method can adapt to an unseen test task using significantly fewer samples during meta-testing than the baselines in 2D-Point Agent and continuous control meta-RL benchmarks, namely, cartpole with unknown angle sensor bias, walker agent with randomized dynamics parameters.

Index Terms—Reinforcement learning, deep learning methods, transfer learning.

I. INTRODUCTION

DESPITE the remarkable achievements of deep reinforcement learning (RL) algorithms [1], [2], they are hindered by poor sample efficiency and limited generalizability. Few-shot meta-reinforcement learning (meta-RL) framework [3], [4] aims to overcome these limitations by learning a distribution of tasks during meta-training and adapting to a new task from the same distribution using a few samples from the new task in meta-testing. Given the high cost associated with collecting data in the real-world, minimizing the number of online interactions for rapid adaptation is essential for real-world applications.

Fast adaptation under uncertainty has various applications in the real-world, such as autonomous driving, disassembly with manipulation robots, medical diagnosis, and ventilation robots. Because the real-world is non-stationary, an agent should be robust to changing environmental dynamics, such as failures in

the sensors and actuators [5]. Relying on deep RL agents trained on a single task can lead to failures in these settings. Thus, meta-RL [4] emerges as a promising direction for increasing robustness through fast adaptation to an unknown test task. However, since a meta-RL algorithm requires samples from the test task for adaptation, it is limited by the number of samples collected from the unknown test task. Increasing sample efficiency reduces dependence on the costly and potentially dangerous process of collecting real-world experiences in an unknown task. This challenge is compounded when the reward signal from the test task is either unavailable or unreliable due to issues such as sensor malfunctions [3], [6]. To increase sample efficiency during meta-testing in settings with unknown rewards, we need to learn an expressive meta-model during training for cost-effective sample generation.

This work investigates meta-RL settings where the transition function changes across a distribution of tasks and the reward signal is missing during meta-testing. Each transition function depends on a hidden dynamics parameter during meta-training and meta-testing. Imagine a setting where an agent is tasked with moving to a goal location in different environments parameterized by different force fields. Different force fields push the agent in different directions; requiring the agent to move robustly with different transition dynamics using few trial-and-error attempts. Similarly, in instances of a sensor malfunction, an unknown bias can result in inaccurate signals, necessitating the robot to adapt to this new environment with minimal damage [6], [7]. In these cases, sample efficiency and not relying on reward signals are crucial, especially when engaging with a novel scenario, referred to as the "test task" where the number of real-world interactions is limited. No-reward Meta Learning [6] is an unsupervised meta-testing algorithm that adapts without reward signals using a learned advantage function. Similarly, Context-aware Dynamics Model (CaDM) [8] and Rapid Motor Adaptation (RMA) [9] adapts within an episode without relying on rewards during meta-testing, though they use fixed length of transitions during training and testing.

We introduce a novel hybrid meta-RL formulation that decouples task inference from parameterized policy gradient-based (PPG) meta-RL [3], [4]. Our approach diverges from prior works [10], [11] by targeting meta-testing sample efficiency through a modular approach while not requiring reward information during meta-testing. This strategy substantially improves sample efficiency in meta-testing by predicting next states given target states and actions queries, thereby generating cost-effective samples from the meta-model rather than

relying on costly online environment interactions. Furthermore, Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP) allows reusing samples collected from PPG meta-RL meta-training for task inference in an offline manner, without compromising sample efficiency during meta-training. This decoupling enhances the flexibility and applicability of our approach in scenarios where retraining the meta-policy with online interactions to improve meta-testing sample efficiency is not feasible.

The key contributions of our method, UMCNP, include (1) improving sample efficiency via fast adaptation to the unseen test task (requires only a single rollout from the unseen test task compared to 25 rollouts used in prior works [6]), (2) generating and learning meta-dynamics models with no access to test task parameters and rewards while being permutation invariant and agnostic to the number of samples, (3) utilizing offline datasets for task inference in meta-RL with unsupervised meta-testing, and (4) decoupling PPG-based and task inference-based meta-RL, offering flexibility of application and enhancing informative trajectory generation. We evaluate our method using a point agent, a cartpole with sensor bias, and the widely used Walker Agent Environment featuring variable dynamics. Our results demonstrate a remarkable improvement in sample efficiency over the unsupervised meta-testing baselines.

II. RELATED WORK

Meta-learning provides a framework for learning new tasks more efficiently from limited experience by using knowledge from previously learned tasks [4], [12]. Meta-learning has demonstrated promising results in regression, classification, and deep RL [3], [4], [6], [13], [14]. Our work builds on the meta-RL framework, where a distribution of tasks shares a common structure but differs in the transition dynamics model. The assumption of a common structure allows unsupervised meta-testing in the new task with unknown transition dynamics model parameters.

A. Black Box Few-Shot Meta-RL

Black box few-shot meta-RL methods assume less structure on the meta-learning algorithm compared to parameterized policy gradient few-shot meta-RL methods while sacrificing generalization [3]. Prior work on black box meta-RL learns the task-specific context variable from a set of experiences collected by a control policy. Notably, Mishra et al. [15], use memory-augmented models like temporal convolutions and soft attention to learn a contextual representation. Within the domain of meta-imitation learning, Duan et al. [16] encode expert demonstrations into a learned context embedding where policy is conditioned on.

B. Parameterized Policy Gradient Based Few-Shot Meta-RL

Parameterized policy gradient-based few-shot meta-RL methods [3], [4], [17], [18], [19], utilize policy gradient methods in the inner loop [3]. For instance, in Model-Agnostic Meta-Learning (MAML) [4], a parameterized policy gradient meta-learning algorithm is used where train and test tasks are sampled from the same distribution. No-reward Meta-Learning (NORML), proposed by Yang et al. [6], is an extension of MAML-RL framework [4] for unsupervised meta-testing where the environment dynamics model changes across tasks instead of the reward function. NORML aims to utilize past experience to quickly adapt to tasks with unknown parameters from a few samples without reward signals. While these methods have demonstrated promising results, they fall short in sample efficiency in meta-testing. We instead leverage the conditional neural processes [20] to reduce the number of experiences an agent needs for adaptation to the test task.

C. Few-Shot Meta Reinforcement Learning With Task Inference

Few-shot meta-RL methods that use task-inference attempt to infer the underlying MDP of the unknown test task in the meta-RL inner loop [3]. Zintgraf et al. [11] uses a recurrent neural network (RNN) architecture to encode posterior over ordered transitions. The posterior belief is used to augment the state variable on which the policy is conditioned. Unlike FOCAL [21], which employs a deterministic encoder, PEARL [10], and VARIBAD [11] use a stochastic encoder. Latter methods train the inference model by optimizing the evidence lower bound (ELBO) [22] objective. FOCAL, on the other hand, clusters transitions in the latent space using an inverse power loss distance metric learning objective under the assumption that tasks are deterministic and there exists a one-to-one function from the transitions to the $T \times R$ where T and R denote the task-specific transition function and reward function, respectively. In contrast to these approaches, while the goal of UMCNP is to identify the latent task parameter in the inner loop, we target few-shot meta-RL with unsupervised meta-testing where the reward information is missing. Moreover, our method decouples task inference from PPG in meta-RL, allowing flexibility. This separation improves sample efficiency in meta-testing even after training has concluded, as long as the samples from meta-training are accessible.

Rapid Motor Adaptation (RMA) [9], categorized as a model-free task inference meta-RL approach [3], have shown remarkable performance in sim-to-real applications on an A1 robot. RMA initially trains a base policy that takes the latent extrinsics vector encoded by the environment factor encoder, the state, and the previous action. Then, it trains an adaptation module to predict latent extrinsics from a history of state-action tuples. During deployment it uses the adaptation module and the base policy since the environment factors are hidden during testing. In contrast to the previous methods, RMA can adapt to the test task at test time within an episode. Although UMCNP can also adapt within an episode due to the flexibility to the number of samples used in training, it does not require environment parameters during training, which may not be available in a robotics dataset, and is only trained with transitions.

Model based meta-RL algorithms focus on learning the dynamics and the reward function of the environment. While they are sample efficient and enjoy the advantages of supervised learning, they can have lower asymptotic performance [3].

CaDM [8] encodes a fixed number of transitions and, similar to RMA, can adapt within an episode. It can be combined with model-free RL by using the latent vector concatenated to the states to train a model-free policy gradient algorithm. CaDM trains the dynamics model and the context encoder by predicting forward and backward dynamics from a fixed number of state and action transitions. Unlike CaDM, UMCNP handles backward and forward state prediction in a permutation-invariant manner, agnostic to the number of observations, offering flexibility during testing without the need to train for varying future and history lengths.

D. Conditional Neural Processes

Conditional Neural Processes (CNPs) [20] predict the parameters of a probability distribution conditioned on a permutation invariant prior data and target input. CNPs aim to represent a family of functions using Bayesian Inference and the high representational capacity of neural networks [23], [24], [25], [26]. Prior approaches have leveraged the robust latent representations obtained by the CNP model in high-dimensional trajectory generation [27], long-horizon multi-modal trajectory prediction [27], [28], and sketch generation tasks [29].

III. PRELIMINARIES

A. Notation

We denote tasks as Markov Decision Processes (MDP). An MDP is represented by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$, and $\mathcal{A}$ are the state and action spaces, $\rho_0(s_0)$ is the initial state distribution, r is the reward function, and $\gamma \in [0, 1)$ is the discount factor [30]. $P(s_{t+1}|s_t, a_t)$ is the state transition dynamics where (s_t, a_t) are the state and action at timestep t, and s_{t+1} is the next state.

The objective of an RL agent is to maximize the expected cumulative discounted reward $\mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t)]$ by learning a policy $\pi_\theta(a_t|s_t)$, parameterized by θ. Advantage function of a state and action tuple $A^\pi(s_t, a_t)$ can be formulated as $A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t)$ where Q-function and value function are $Q^\pi(s_t, a_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}[\sum_{k=0}^\infty \gamma^k r_{t+k}]$, and $V^\pi(s_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}}[\sum_{k=0}^\infty \gamma^k r_{t+k}]$, respectively [31]. Advantage function estimates the relative advantage of taking an action in that state with respect to the estimated value of that state [32].

B. Model-Agnostic Meta-Learning

In meta-RL, we learn to adapt to a distribution of tasks represented by $p(T)$, where each task is distinguished by a unique Markov Decision Process (MDP) despite sharing a common structure. Typically, the tasks share identical state and action spaces, yet they vary in terms of their reward functions and state transition dynamics. A prominent approach for gradient-based meta-RL is the Model-Agnostic Meta-Learning (MAML) approach [4]. During meta-training, MAML trains a meta-policy π_θ, also known as an initial or a base policy [3], characterized by parameters θ. To adapt quickly to a new task from this distribution, the meta-policy requires only a small number of

samples, $\mathcal{D}_i^{tr}$ following the equation

$$\phi_i = \theta + \alpha \nabla_\theta \hat{\mathcal{J}}(\theta, \mathcal{D}_i^{tr}) \tag{1}$$

where α is the adaptation learning rate, ϕ_i is the adapted policy parameter, and $\hat{\mathcal{J}}(\theta, \mathcal{D}_i^{tr})$ is the estimated RL objective for task T_i [4]. After applying this adaptation step separately to a batch of tasks in the inner loop, we obtain $\mathcal{D}_i^{test}$ from each task with the adapted parameters. Subsequently, we compute the meta-objective in the outer loop $\max_\theta \sum_{T_i \sim p(\mathcal{T})} \hat{\mathcal{J}}(\phi_i, \mathcal{D}_i^{test})$:

$$\max_\theta \sum_{T_i \sim p(\mathcal{T})} \hat{\mathcal{J}}(\theta + \alpha \nabla_\theta \hat{\mathcal{J}}(\theta, \mathcal{D}_i^{tr}), \mathcal{D}_i^{test}) \tag{2}$$

with respect to the meta policy parameters θ [3], [4]. Once the meta-training is finished, the agent collects a few samples in the new test task during meta-testing and adapts its meta policy using (1).

C. No-Reward Meta Learning

Provided that the change in dynamics can be represented in a set of tuples of state (s), action (a), and next-state (s') from the same task $\{(s, a, s')\}$, NORML learns a pseudo-advantage function $A_\psi(s, a, s')$ [6]. It is important to note that the aim of A_ψ is to guide the meta-policy adaptation in (1) instead of fitting to the advantage function, thus we refer to the learned advantage function as the pseudo-advantage function. Task-specific gradient update in NORML, $\phi_i = \theta + \theta_{\text{offset}} + \alpha \nabla_\theta \hat{\mathcal{J}}_{\mathcal{T}_i}^{\text{NoRML}}(\theta, D_i^{\text{train}})$, is computed [6] with

$$\theta + \theta_{\text{offset}} + \alpha \sum_{D_i^{\text{train}}} A_\psi(s, a, s') \nabla_\theta \log \pi_\theta(a \mid s) \tag{3}$$

where A_ψ is used to compute task-specific parameters in the meta-learning inner loop from a set of state, action, and next-state transitions of task i denoted as D_i^{train} that does not contain reward signals. In this adaptation phase, the pseudo-advantage function guides the policy gradient update. To encourage decoupling between the meta policy and the task-specific policy in the inner loop, we learn an offset parameter θ_{offset} end-to-end [6]. This offset parameter, added in the inner loop adaptation formulated in (3), is identical for all tasks.

The learned pseudo-advantage function A_ψ is optimized in the meta-learning outer loop. Using D_i^{train}, and D_i^{test}, we not only update ψ and θ in the outer loop, but additionally update the offset parameters θ_{offset} [6]:

$$\begin{bmatrix} \theta \\ \theta_{\text{offset}} \\ \psi \end{bmatrix} \leftarrow \begin{bmatrix} \theta \\ \theta_{\text{offset}} \\ \psi \end{bmatrix} + \beta \sum_{T_i \sim \mathcal{T}} \nabla_{[\theta^T \theta_{\text{offset}}^T \psi^T]^T} \hat{\mathcal{J}}_{T_i}(\phi_i, D_i^{\text{test}}) \tag{4}$$

where $\nabla \hat{\mathcal{J}}_{T_i}(\phi_i, D_i^{\text{test}})$ is the gradient of the meta objective formulated as $\sum_{D_i^{\text{test}}} A^\pi(s, a) \nabla \log \pi_{\phi_i}(a|s)$, $A^\pi(s, a)$ is the RL advantage function, and β is the learning rate.

D. Conditional Neural Processes (CNP)

The architecture of CNP consists of parameter-sharing encoders and a decoder named the query network [20]. Initially,

a set of random input and output pairs $(x_{f^i}, y_{f^i}^{true})$ are sampled from a function $f^i \in F$. These pairs are then encoded into latent representations through parameter-sharing encoder networks. Subsequently, the average of these representations is computed, further ensuring that the model is invariant to permutations and the number of inputs. The averaged representation is then concatenated with the target input query and passed to the query network. Finally, the query network generates the predicted mean and standard deviation for the queried input $(x_{f^i}^q)$.

IV. PROPOSED METHOD: UNSUPERVISED META-TESTING WITH CONDITIONAL NEURAL PROCESSES (UMCNP)

In this section, we present our method, Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP), for learning dynamics models by leveraging meta-training trajectories collected by PPG meta-RL. By sampling actions from the policy, the UMCNP model facilitates the autoregressive generation of complete trajectories. This process involves querying the model with the action and the state, receiving a prediction of the next state, and then feeding this state back into the model alongside new actions. A learned pseudo-advantage function takes state, action, and next state as input and guides the policy gradient in the adaptation. As a result, UMCNP effectively creates models of unknown environments with fewer samples, a significant advancement we detail after outlining the problem context.

Our method aims to enhance sample efficiency in meta-RL with unsupervised meta-testing. UMCNP requires fewer samples from the test task to generate trajectories without accessing environment parameters or rewards, addressing the challenge of unsupervised meta-testing. This procedure facilitates the generation of cost-effective trajectories for update during meta-testing.

a) Gradient-Based Meta Reinforcement Learning with Unsupervised Meta-Testing: During meta-training, we store transitions for each $task_i$ as a set of transition tuples $B_i = \{(s, a, s')\}_{t=0}^{H-1} \subset S \times A \times S$ without the task parameter and the rewards. We then use these batches to create our offline dataset illustrated in Fig. 1.

Both $A^\pi(s, a)$ and the pseudo-advantage function use polynomial regression with a fitted value function [6]. Policy parameters in the inner loop adaptation in (3) and the outer loop meta-optimization in (4) are updated using vanilla PG and actor-critic Proximal Policy Optimization [1] method with a fitted value function, respectively. We use Adam optimizer [33] for both the inner loop and the [6] outer loop update. After we learn the meta-policy, offset parameter $\boldsymbol{\theta}_{offset}$ and the pseudo-advantage function $A_\psi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ using NORML, we train our UMCNP model using the offline dataset of transitions $D = \{B_i\}_{i=1}^n$ where n denotes the number of transitions.

b) Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP): We train the UMCNP offline using unlabeled batches of transition tuples we previously collected during the first phase of online PPG meta-training. Fig. 1 illustrates the second phase of the meta-training procedure for UMCNP directed towards task inference. Notably, the transition dynamics parameter is hidden, and the data does not contain reward information during training and testing. In each meta-training iteration, we
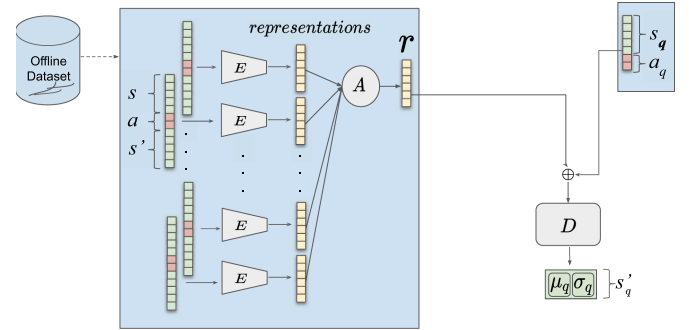

Fig. 1. Meta-training procedure of our proposed method UMCNP. Randomly selected samples from a task are extracted from the offline dataset and encoded into fixed-size representations. An averaging module A computes the mean of these representations, denoted by $\mathbf{r}$. This mean representation, $\mathbf{r}$, is then concatenated with the state query, and action query from the same task $[s_q, a_q]$, also drawn from the offline dataset, are fed to the decoder network as $[s_q, a_q, r]$. The decoder then outputs the mean μ_q and standard deviation σ_q for the next state $\mathbf{s}'$ conditioned on the $[s_q, a_q, r]$ input vector.

randomly select an unlabeled batch $B_i = \{(s, a, s')\}_{t=0}^{H-1}$ from the offline dataset D. From this batch, we randomly select a set of task-specific transition tuples $\{(s_k, a_k, s'_k)\}_k$ and a target query tuple (s_q, a_q, s'_q) without replacement. We utilize (s_q, a_q, s'_q) for the target state-action query $[s_q, a_q]$ and true target next state label $[s'_q]$.

Next, we encode each tuple (s_k, a_k, s'_k) into a fixed size representation using a parameter sharing encoder network. These representations are then passed through an averaging module A to obtain a learned latent vector r that represents the hidden environment transition function parameter used in batch B_i. We then concatenate the resulting latent representation r with the $[s_q, a_q]$ to predict the distribution parameters μ_q, σ_q of the next state s'_q conditioned on r and the target query $[s_q, a_q]$:

$$\mu_q, \sigma_q = f_{\theta_D}\left([s_q, a_q] \oplus \frac{1}{k}\sum_k g_{\theta_E}(s_k, a_k, s'_k)\right) \quad (5)$$

where f_{θ_D}, g_{θ_E} are the decoder and the encoder networks, (s_k, a_k, s'_k) are the randomly sampled transitions from the set B_i. The loss function of UMCNP can be expressed as

$$\mathcal{L}(\theta_E, \theta_D) = -\log P\left(s_q'^{true} \mid \mu_q, softplus(\sigma_q)\right). \quad (6)$$

The softplus activation function is applied to σ_q, and the resulting output, along with μ_q, are used to compute a multivariate normal distribution with a diagonal covariance. A high variance in this distribution can indicate high uncertainty in the predicted next state value.

During meta-testing in Fig. 2, an agent collects a few samples $\{(s_k, a_k, s'_k)\}$ from the unseen test task without rewards. A trained encoder network with shared weights encodes each sample into a fixed-size representation. Fig. 2(I) shows how an averaging module (A) forms a learned shared representation of the test task dynamics. Using this shared latent vector representing the hidden task parameter, we can generate cost-effective rollouts using our meta-policy. This representation helps in
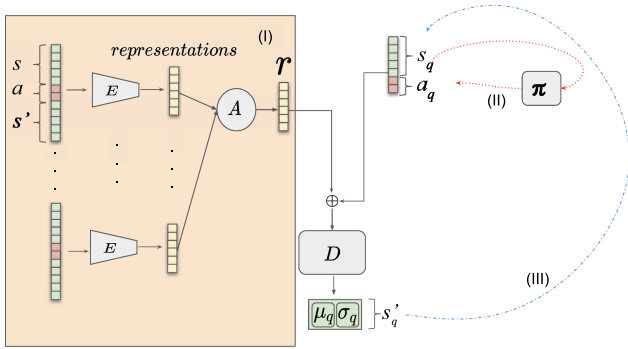
Fig. 2. Meta-testing procedure of our proposed method UMCNP. The meta-policy is allowed to collect a few samples from the test task. (I) These samples ($\{(s_k, a_k, s'_k)\}$) are encoded into fixed-size representations and averaged using module A. (II) Then, the meta-policy samples the action query a_q conditioned on the initial state query s_q to obtain $[s_q, a_q]$, which is concatenated with the mean representation **r** before being input to the decoder network. The decoder's predicted next state is used to sample the meta-policy's next action query. Through this method, we sequentially construct a complete trajectory, leveraging the dynamic modeling capabilities of our model.

predicting the parameters of the next-state distribution with the state-action query $[s_q, a_q]$.

For every rollout generated by UMCNP, we use the same true initial state sampled from the true test task. We input this initial state into the stochastic meta-policy to obtain the action query a_q depicted in Fig. 2(II). Due to the stochasticity of the meta-policy trained with PPO, we can generate diverse rollouts from our UMCNP model which can facilitate informative exploration for our meta-policy. We then take the predicted next state to determine the subsequent action query in Fig. 2(III), repeating the process of (III–II) and querying the decoder until the episode concludes. We combine these rollouts generated from UMCNP with the ground-truth rollout sampled from the test task.

The learned pseudo-advantage network takes a tuple (s, a, s') as input and outputs an advantage estimation value. Hence, the pseudo-advantage network $A_\psi(s, a, s')$ computes advantage estimates for each tuple in the combined set of generated roll-outs and a ground-truth rollout. Finally, we perform a gradient update on the meta-policy for fast adaptation to the test task. This update uses the estimated advantage values, the learned parameter offset, and the ensemble of rollouts generated by UMCNP alongside a single rollout from the test task. Although the reward information is completely missing in meta-testing and only available during the first phase of meta-training, our model is capable of remarkably sample-efficient unsupervised adaptation.

V. EXPERIMENTS

In this section, we analyze the performance of our method and compare it with meta-RL baselines in 2D point agent and continuous control tasks, including cartpole with sensor bias and biped locomotion of a walker agent with random dynamics parameters. ORACLE agent, trained using NORML, serves as a baseline to demonstrate the potential performance when extensive sampling from the test task is allowed. Unlike ORACLE [6], UMCNP,

NORML [6], Probabilistic Ensembles with Trajectory Sampling CaDM (PE-TS CaDM), Vanilla CaDM [8], PE-TS [34], Proximal Policy Optimization-CaDM (PPO-CaDM) [8], Vanilla DM [8] have limited access to the test task during meta-testing. Despite this limitation, UMCNP achieves fast adaptation to the test task with significantly less interaction with the test task than NORML, matching ORACLE's performance. In all our experiments, we report results with 95% confidence intervals with algorithms trained with 5 random seeds.

A. 2D Point Agent With Unknown Artificial Force Field

The goal of the point agent, initialized at $(x = 0, y = 0)$, is to move to the position $(x = 1, y = 0)$, where $(x, y) \in [-2, 2] \times [-2, 2]$ are the positions on the 2D plane. We are interested in a meta-RL setting used in [6] where the reward function is identical across multiple tasks. Different tasks are created by generating different artificial force fields that push the agent in different directions using the parameter (ω). Dynamics of the 2D point task are formulated as:

$$\begin{bmatrix} s_{t+1}^{(1)} \\ s_{t+1}^{(2)} \end{bmatrix} = \begin{bmatrix} \cos \omega & -\sin \omega \\ \sin \omega & \cos \omega \end{bmatrix} \begin{bmatrix} a_t^{(1)} \\ a_t^{(2)} \end{bmatrix} + \begin{bmatrix} s_t^{(1)} \\ s_t^{(2)} \end{bmatrix}$$

where state vector denoted by $[s_t^{(1)}, s_t^{(2)}]^T$ corresponds to the x, y-coordinates on the 2D plane, rotation parameter is defined by ω, and the action vector at time t, $[a_t^{(1)}, a_t^{(2)}]^T$, refer to the movement in the x,y-directions with $(dx_t, dy_t) \in [-1, 1]$ [6]. The maximum episode length in this environment is 10. We use the same reward function, the negative Euclidean distance from the goal position, and hyperparameters in NORML for comparative analysis. In the Point Agent environment, 5000 tasks are defined over the $[-\pi, \pi]$ interval. The agent is initially trained across a distribution of 5000 tasks, i.e. in environments with 5000 different force fields. Then, it is tested in unseen test tasks. The rollouts obtained from direct interactions with the test task will be referred to as actual rollouts. We have also trained our UMCNP model with the advantage values corresponding to the state, action, next state tuples recorded during meta-training and we name this variation as UMCNP-ADV. UMCNP-ADV takes in state, action, next state, and advantage tuples and predicts the mean and standard deviation of the next state and the advantage for the state, action target query. At meta-test time, the ORACLE agent is allowed to collect 25 actual rollouts from the unseen test task. Conversely, NORML agent, UMCNP and UMCNP-ADV are allowed to use a single actual rollout for fine-tuning the meta-policy. For a reliable evaluation, we use the same actual rollout data for UMCNP and UMCNP-ADV models that is used for the single rollout NORML fine-tuning. Fig. 3 illustrates the trajectories followed by the adapted policies where the trajectories of UMCNP and ORACLE agents are similar. Fig. 4 shows the post-update cumulative reward obtained in the test tasks for ORACLE, UMCNP, NORML, and UMCNP-ADV with blue, orange, green, and red respectively. The results in Table I show the mean with 95% confidence intervals and median post-update cumulative rewards collected in the test tasks. Adaptation using 24 generated rollouts from the UMCNP model achieves -1.02
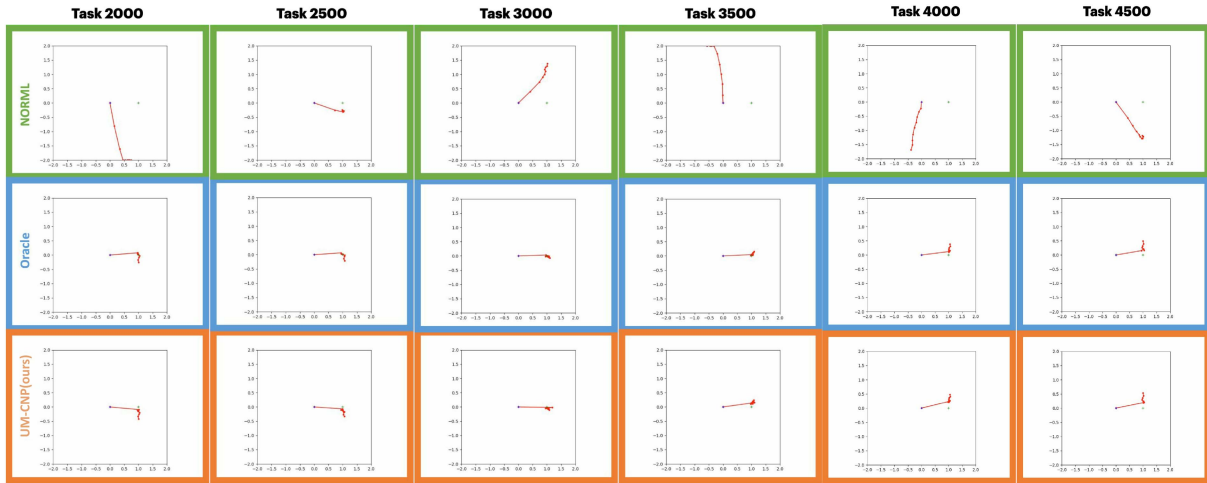
Fig. 3.   Test task indices 2000, 2500, 3000, 3500, 4000, and 4500 correspond to task parameter ω with values $-2\pi/10$, 0, $2\pi/10$, $4\pi/10$, $6\pi/10$, $8\pi/10$ respectively.
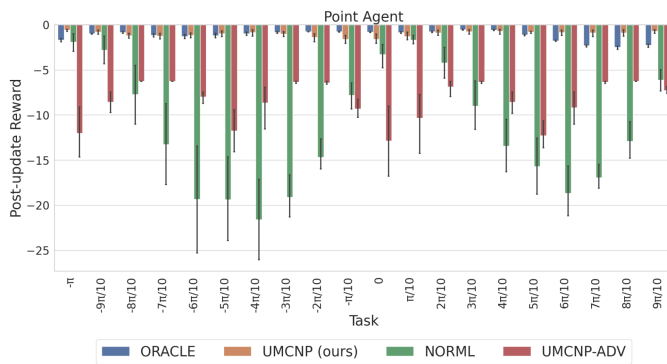

Fig. 4.   Point agent plot shows the post-update cumulative reward (sum of rewards) (y-axis) in test tasks, computed after finetuning with 24 generated rollouts and 1 actual rollout for UMCNP (ours) (orange), 1 actual rollout for NORML [6] (green), and 25 actual rollouts for ORACLE/Ground-truth (blue) and UMCNP-ADV (red) with 95% confidence intervals.
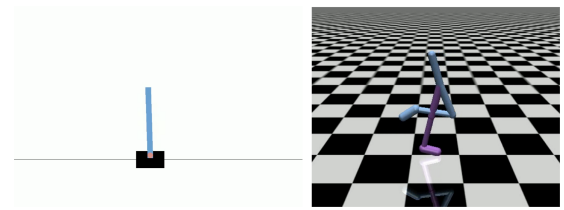

Fig. 5.   Visualization of cartpole with unknown angle sensor bias environment (left), walker-2d agent environment with random dynamics parameters (right) environments.

TABLE I
2D POINT AGENT ENVIRONMENT POST-UPDATE REWARD WITH 95%
CONFIDENCE INTERVALS

Point Agent	mean	median
UMCNP(ours)	**-1.02 $\pm$ 0.09**	**-0.91**
NORML	-11.49 $\pm$ 1.44	-11.01
UMCNP-ADV	-8.49 $\pm$ 0.58	-7.46
ORACLE	-1.20 $\pm$ 0.12	-0.94

$\pm$ 0.09, similar to the ground truth ORACLE with (25 actual rollouts) -1.20 ± 0.12, whereas NORML underperforms with -11.49 ± 1.44. Although UMCNP-ADV performs better than NORML, we observe that the UMCNP is significantly better than UMCNP-ADV. Future work will explore architectural improvements, such as separate decoders, to enhance UMCNP-ADV's performance.

B. Cartpole With Unknown Angle Sensor Bias

We evaluate our method on the Cartpole with Angle Sensor Bias environment (Fig. 5) [6], an extension of the Cartpole

environment simulated in MuJoCo, Gym [35], [36]. In this experiment, tasks differ by a hidden position sensor drift parameter in the range $[-8°, 8°]$. In meta-training, tasks are sampled from a uniform distribution where the hidden variable parameterizes the dynamics function. The state space consists of the position/angle and velocity/angular velocity of the cart and the pole. To evaluate the adaptation performance where a limited number of transition tuples are available when the maximum episode length of the environment is 500, we compared the performance of UMCNP to NORML with only 5 and 50 transitions, and PE-TS CaDM [8], Vanilla CaDM [8], PPO-CaDM [8], trained with history-length and future transition length of 5 and Random Shooting with 1000 candidates for planning following the Cartpole implementation details in [8], PE-TS [34], Vanilla DM [8] by using the code available in [37]. Importantly, CaDM requires access to the reward function during training for Model Predictive Control [38] and we have provided the reward function definitions during training. Consistent with other experiments, results in Fig. 6 and Table II indicate that UMCNP outperforms the baselines and CaDM performs better when integrated in model-free RL.

C. Walker-2D With Random Dynamics Parameters

In order to examine the sample efficiency of UMCNP in complex, higher dimensional control tasks, we evaluate our method in the Walker-2D Agent Environment with random dynamics parameters [13] with a maximum episode length of 1000in an
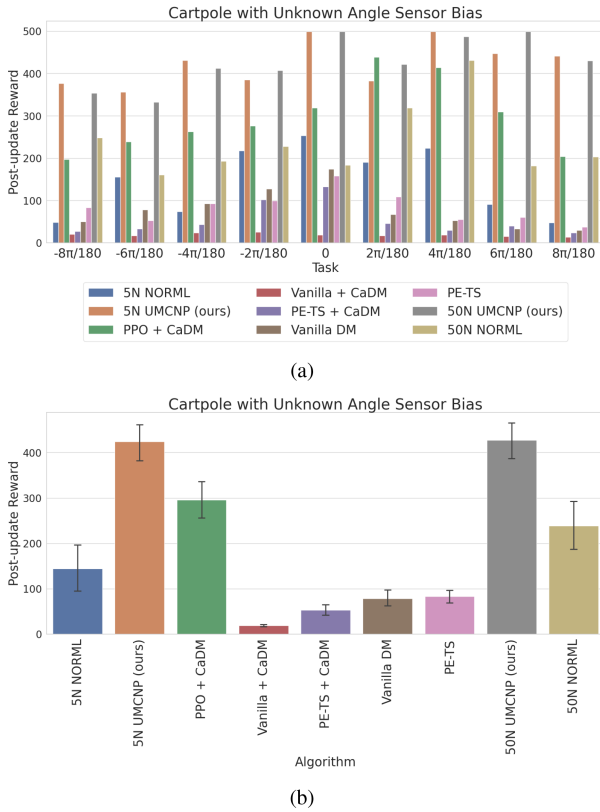
(a)

(b)

Fig. 6. (a) Post-update cumulative Reward (y-axis) in test tasks compared with baselines. (b) Overall performance compared with baselines with 95% Confidence Intervals. 5N UMCNP (ours), 5N NORML and 50N UMCNP (ours), 50 N NORML uses 5 and 50 test task transitions respectively. X-axis represents the test task parameter.

TABLE II
POST-UPDATE CUMULATIVE REWARD OF UMCNP AND THE BASELINES

| Cartpole Agent	mean	median
5N NORML	144.56 ± 48.72	45
5N UMCNP (ours)	424.13 ± 39.79	**499**
PPO + CaDM	295.40 ± 39.07	280
Vanilla + CaDM	18.62 ± 2.32	16
PE-TS + CaDM	52.82 ± 12.36	35
Vanilla DM	78.22 ± 16.87	57
PE-TS	82.87 ± 14.15	69
50N UMCNP (ours)	$\mathbf{427.02} \pm 42.76$	**499**
50N NORML	238.51 ± 55.59	174
Walker − 2D Agent	**mean**	**median**
ORACLE	1755.07 ± 370.89	1303.97
UMCNP (ours)	$\mathbf{1844.62} \pm 338.80$	**1602.97**
NORML	1608.97 ± 316.83	1135.64
10N UMCNP (ours)	1684.56 ± 355.71	1072.68
10N NORML	1566.38 ± 308.84	1203.82
PPO + (Vanilla + CaDM)	171.18 ± 57.00	199.05
Vanilla DM + CaDM	22.31 ± 17.79	-7.28
PE-TS + CaDM	-1.20 ± 1.86	-2.47
Vanilla DM	-4.34 ± 0.37	-4.47
PE-TS	-0.48 ± 1.45	-2.42
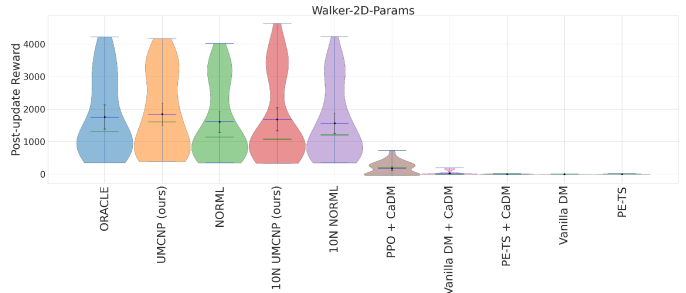
Fig. 7. Violin plots of post-update cumulative reward from unseen 10 meta-test tasks with 95% Confidence Intervals (black). These plots combine vertically aligned kernel density plots that represent the data distribution, with markers clearly indicating the minimum, maximum, mean (blue), and median (green) values. 10N UMCNP (ours) and 10N NORML uses 10 test task transitions respectively.

unsupervised meta-testing setup where the agent receives no reward signal from the test task. Different locomotion tasks are created by unknown body mass, body inertia, damping on different joints, and friction parameters. The scaling parameters for body mass, body inertia, and friction parameters are sampled from a range determined by raising 1.5 to powers uniformly distributed between -3 and 3, in the interval $[1.5^{-3}, 1.5^3]$. Likewise, the scaling parameters for damping across different joints follow the same methodology but use a base of 1.3 covering the range $[1.3^{-3}, 1.3^3]$. Walker environments with slimmer distributions, for instance only parameterizing the body mass or reducing the interval of the uniform distributions does not show a significant increase in performance for NORML compared to domain randomization. These scenarios do not require sophisticated adaptation methods; hence, we use more established meta-RL benchmarks [6], [13].

We use only 40 tasks for meta-training sampled from a uniform distribution and compare our method with the baselines on 10 test tasks similar to the meta-testing experiments with rewards [10]. Violin plots in Fig. 7 show the post-update cumulative reward in meta-testing, where NORML, UMCNP use a single rollout, 10N NORML, 10N UMCNP use 10 transitions, and PE-TS CaDM [8], Vanilla CaDM [8], PPO-CaDM [8] with normalization and Cross Entropy Method [39] for planning following [8] use a history and future transition length of 10, and

the ORACLE uses 25 rollouts for adaptation. Results in Table II indicate that UMCNP, not only improved sample efficiency but has the potential to improve meta-test adaptation performance for a larger range of unseen tasks. One reason for that can be the bias-variance trade-off in noisy real samples and generated samples. UMCNP network is trained on the meta-train dataset, hence it can generate less noisy and potentially more robust samples that are seen during training for the meta-policy which can make the adaptation more robust.

VI. CONCLUSION

We propose Unsupervised Meta-Testing with Conditional Neural Processes, a new meta-RL with unsupervised meta-testing method that can generate useful rollouts for sample-efficient meta-policy adaptation. The agent performs without the need for rewards during meta-testing and learns without explicit task parameters, depending entirely on interaction data, during both meta-training and meta-testing. Our results indicate that our method's meta-test performance, when fine-tuned with generated data, shows significantly better performance in

point and cartpole environments, and better performance in walker environments. Crucially, its performance is similar to the ground-truth method, which is allowed to interact with the environment significantly more. This sample efficiency across all experiments is achieved through the generation of samples from fewer number of transition tuples.

In conclusion, we show that leveraging generated data for meta-updates not only improves sample efficiency but can also enhance post-update performance. Additionally, the successful application of CNPs to high-dimensional inputs, such as image data [27], suggests promising avenues for future work. Specifically, applying our findings to high-dimensional sensorimotor spaces, such as manipulator robots equipped with RGB-D cameras, is an exciting direction for further exploration.

APPENDIX

Our model features a 128-dimensional representation and two hidden layers, each with 128 units, in both the encoder and decoder. It is trained with a batch size of 32 with 50 input tuples and 50 target query tuples for 500000 iterations with Adam optimizer [33] with a learning rate of 0.0001. Additionally, the model utilizes the last 10% of trajectories from D^{train} which are gathered by the meta-policy throughout the meta-training phase.

ACKNOWLEDGMENT

REFERENCES

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: https://openai.com/index/openai-baselines-ppo/

[2] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[3] J. Beck et al., "A survey of meta-reinforcement learning," 2023, *arXiv:2301.08028*.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[5] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 10237–10257, 2024, doi: 10.1109/TNNLS.2023.3250269.

[6] Y. Yang, K. Caluwaerts, A. Iscen, J. Tan, and C. Finn, "NoRML: No-reward meta learning," in *Proc. 18th Int. Conf. Auton. Agents Multiagent Syst.*, N. Agmon, M. E. Taylor, E. Elkind, and M. Veloso, Eds., Montreal, Canada, May 2019, pp. 323–331.

[7] I. Clavera et al., "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HyztsoC5Y7

[8] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5757–5766.

[9] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst. XVII*, Jul. 2021.

[10] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5331–5340.

[11] L. Zintgraf et al., "VariBAD: A very good method for bayes-adaptive deep RL via meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Hkl9JlBYvr

[12] J. Schmidhuber, "Evolutionary principles in self-referential learning–diploma thesis," *Technische Universität München*, 1987.

[13] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "ProMP: Proximal meta-policy search," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SkxXCi0qFX

[14] B. Liu, "A theoretical understanding of gradient bias in meta-reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022, pp. 31059–31072.

[15] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *Proc. Int. Conf. Learn. Representations*, 2018.

[16] Y. Duan et al., "One-shot imitation learning," *Adv. Neural Inf. Process. Syst.*, I. Guyon et al., Eds., Curran Associates, Inc., vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf

[17] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 31, 2018.

[18] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," 2017, *arXiv:1707.09835*.

[19] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal model-agnostic meta-learning via task-aware modulation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 32, 2019.

[20] M. Garnelo et al., "Conditional neural processes" in *Proc. Int. Conf. Mach. Learn.*, PMLR, Jul. 2018, pp. 1704–1713.

[21] L. Li, R. Yang, and D. Luo, "Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization," in *Proc. Int. Conf. Learn. Representations*, 2020.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Representations*, Apr. 2014.

[23] L.-L. Bao, J.-S. Zhang, and C.-X. Zhang, "Spatial multi-attention conditional neural processes," *Neural Netw.*, vol. 173, 2024, Art. no. 106201.

[24] W. Bruinsma et al., "Autoregressive conditional neural processes," in *Proc. Int. Conf. Learn. Representations*, 2023.

[25] H. Kim et al., "Attentive neural processes," in *Proc. Int. Conf. Learn. Representations*, 2019.

[26] Z. Ye and L. Yao, "Contrastive conditional neural processes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9687–9696.

[27] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, "Conditional neural movement primitives," in *Proc. Robot.: Sci. Syst.*, vol. 10, 2019.

[28] M. Y. Seker, A. Ahmetoglu, Y. Nagai, M. Asada, E. Oztop, and E. Ugur, "Imitation and mirror systems in robots through deep modality blending networks," *Neural Netw.*, vol. 146, 2022, Art. no. 22.

[29] S. E. Ada and M. Y. Seker, "Generative adversarial networks with conditional neural movement primitives for an interactive generative drawing tool," 2021, *arXiv:2111.14934*.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[31] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Representations*, 2016.

[32] S. E. Ada, E. Ugur, and H. L. Akin, "Generalization in transfer learning: Robust control of robot locomotion," *Robotica*, vol. 40, no. 11, pp. 3811–3836, 2022.

[33] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic gradient descent," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[34] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 31, 2018.

[35] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[36] G. Brockman et al., "Openai gym," 2016, *arXiv:1606.01540*.

[37] Y. Seo, "CaDM code repository," 2020. [Online]. Available: https://github.com/younggyoseo/CaDM

[38] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice–a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[39] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of Statistics*, vol. 31. Amsterdam, The Netherlands: Elsevier, 2013, pp. 35–59.