ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230588

Task-Prompt Generalised World Model in Multi-Environment Offline Reinforcement Learning

Xuantang Xiong[a,b], Linghui Meng[b], Jingqing Ruan[b,c], Qingyang Zhang[b,c], Guoqi Li[b], Dengpeng Xing[a,b;*] and Bo Xu[a,b;]**

[a]School of Artificial Intelligence, University of Chinese Academy of Sciences
[b]Institute of Automation, Chinese Academy of Sciences
[c]School of Future Technology, University of Chinese Academy of Sciences

Abstract. Offline reinforcement learning (RL) circumvents costly interactions with the environment by utilising historical trajectories. Incorporating a world model into this method could substantially enhance the transfer performance of various tasks without expensive calculations from scratch. However, due to the complexity arising from different types of generalisation, previous works have focused almost exclusively on single-environment tasks. In this study, we introduce a multi-environment offline RL setting to investigate whether a generalised world model can be learned from large, diverse datasets and serve as a good surrogate for policy learning in different tasks. Inspired by the success of multi-task prompt methods, we propose the Task-prompt Generalised World Model (TGW) framework, which demonstrates notable performance in this setting. TGW comprises three modules: a task-state prompter, a generalised dynamics module, and a reward module. We implement the generalised dynamics module as a transformer-based recurrent state-space model and employ prompts to provide task-specific instructions, enabling TGW to address the internal stochasticity of the generalised world model. On the MuJoCo control benchmarks, TGW significantly outperforms previous offline RL algorithms in multi-environment setting.

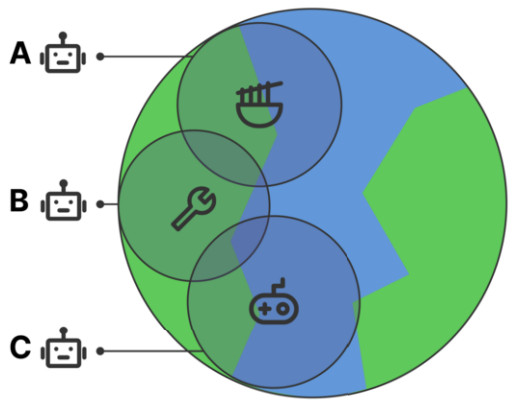

Figure 1. The three robots have different desires, corresponding to the motivation to complete different tasks. Robot A is hungry and receives rewards from eating in the eating environment. Robot B is inspired and receives rewards from outstanding work in the working environment. Robot C is tired and receives rewards from entertainment in the playing environment. They all interact with a subset of the world model.

1 Introduction

Offline reinforcement learning (RL) refers to learning a high-reward policy based on a fixed dataset of previous experiments or human demonstrations, which presents a promising paradigm for data reuse and safe policy learning [15, 17]. This data-driven method holds particular value in domains such as healthcare, autonomous driving, and robotics, as it circumvents costly or dangerous active exploration. While most offline RL methods attempt to address the inherent challenges of extrapolation error [6] and can perform well on a single task, they struggle to generalise to multiple distinct tasks. Moreover, such a task-specific approach reliant on extensive handcrafted engineering is impractical for large-scale use. Although some prior work in the RL field focuses on the multi-task problem, the majority of these efforts are limited to multi-tasks with various reward functions in the same environment [1, 26]. These methods relying on task information embedded within the reward functions, cannot adapt to distinct environments, which hinders their widespread application in society. Intuitively, an agent capable of broad applicability across a wide range of environments is essential for constructing a general-purpose agent, which represents a long-term and crucial goal in the field of artificial intelligence. However, the complexity arising from different types of generalisation has led to this interesting and vital setting receiving limited attention.

In contrast, humans can easily draw from offline experience and adapt it to improve performance in various environments. We believe one of the key reasons is that humans have a generic perception of patterns in the physical world, which can improve data efficiency and the ability to generalise to multiple tasks [16]. Imagine a scenario shown in Figure 1, where the agent has different desires at various periods. Each task requires interaction with a subset of the entire world, with the underlying laws of the world being uniform and self-consistent. Empirically, the model-based RL method is potentially an effective way to enhance generalisation due to the reusability of its dynamics [31]. Therefore, we decided to leverage a model-based method to address the aforementioned multi-environment challenges.

Our goal is to investigate whether a generalised world model can be learned from offline datasets and serve as a good surrogate for

* Corresponding Author. Email: dengpeng.xing@ia.ac.cn
** Corresponding Author. Email: xubo@ia.ac.cn

policy optimisation in multi-environments. Firstly, we introduce a multi-environment offline RL setting, defining its configuration and evaluation metrics. To tackle this problem, we propose a Task-prompt Generalised World Model (TGW) framework, which is a paradigm for building a world model with a single set of parameters from offline data to support agents performing well in diverse environments. As multi-task prompt methods offer notable performance and scaling properties in vision and language [23, 19], we incorporate this architectural inductive bias into the TGW framework.

The TGW framework includes a task-state prompter module for integrating task information, a generalised dynamics module for fitting multiple dynamics, and a reward module for specifing task objectives. Specifically, we implement TGW while considering the properties of constructing a world model in RL. We propose a model named TransRSSM as the generalised dynamics model, an architecture that can tackle the problem of exploiting environments with distinctively different dynamics.

We conduct experiments based on the GYM-MuJoCo environment [2]. Compared to other offline model-based methods, our method has significant advantages in all multi-environment tasks, especially when the offline data is not sampled by an expert policy. In addition, TGW also demonstrates successful and effective adaptation to unseen tasks. Our contributions are listed below:

1. We propose a novel multi-environment offline RL setting and a TGW framework.

2. We propose TransRSSM, which combines the advantages of transformers and recurrent state-space models.

3. We experimentally investigate the performance of TGW in seen and unseen tasks, supporting the conclusion that it is capable of handling multi-environment tasks.

2 Preliminary

Partially Observable Markov Decision Process (POMDP) can be defined as a tuple $\mathcal{M} = \{S, A, O, r, P, \rho_0, \gamma\}$, where an agent is in a state $s_t \in S$, gets an observation $o_t \in O$ derived from an observation function $O(o_t|s_t)$, and decide to take an action $a_t \in A$ at time step t. The policy of agent can be represented as $\pi(a_t|o_t)$. And then s_{t+1} will be arrived according to the state transition function $P(s_{t+1}|s_t, a_t)$, the agents will receive a new observation o_{t+1} and a reward $r(s_t, a_t, s_{t+1})$. In addition, ρ_0 represent the distribution of initial state s_0, and γ is discounted factor. Typically the goal of reinforcement learning is to find the optimal policy $\pi^*(a_t|s_t)$, that can maximize the expected sum of discounted reward donated by $\eta = E_{\pi,\mathcal{M}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

In the **Model-Based RL (MBRL)**, an agent can learn or predict with the help of a model without interacting with the real environment. A learned POMDP model can be represented as $\hat{\mathcal{M}} = \{S, A, O, \hat{r}, \hat{P}, \hat{\eta_0}, \gamma\}$. In this expression, the reward function $\hat{r}$, the transition function $\hat{P}$, and the initial state distribution $\hat{\eta_0}$ are learned using interactions sampled from the real environment $\mathcal{M} = \{S, A, O, r, P, \eta_0, \gamma\}$.

In the **Offline RL** setting, we have access only to fixed datasets $D = (o_i, a_i, r_i, o_{i+1})_{i=1}^N$ to optimize our policy, where N denotes the length of the datasets. These datasets may be generated by a diversity behaviour policy.

3 Multi-environment Offline RL

In the multi-environment offline RL setting, we aim to leverage multi-environment offline datasets, denoted as $D^{train} = (D^k)_{k=1}^K$, to enable the agent to perform effectively across diverse environments. In this notation, K represents the total number of datasets collected from various environments. A specific task can be represented by a POMDP $\mathcal{M}^k = \{S, A, r^k, P^k, \eta_0^k, \gamma\}$. The offline data $D^k = (s_i, a_i, r_i, s_{i+1})_{i=1}^N$ is gathered by interacting with the environment specified by $\mathcal{M}^k$ using either a pure or a mixed behaviour policy $\pi^k(a_i|s_i)$. For the sake of brevity, we omit O in this context and consider $O(o_t|s_t)$ in the subsequent implementation of our algorithm.

Two distinct evaluation processes are proposed. The first assesses the agent's ability to adapt to tasks encountered within diverse contexts. In other words, agents are trained using the dataset $(D^k)_{k=1}^K$, which is sampled from $\mathcal{M}^{train} = \{\mathcal{M}^k\}_{k=1}^K$, and subsequently evaluated in the environment $\mathcal{M}^{seen} \in \mathcal{M}^{train}$. The objective is to maximise $\eta = E_{\pi,\mathcal{M}^{seen}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The second process evaluates the agent's generalisation capacity for unseen tasks, implying that the agent is assessed on $\mathcal{M}^{unseen}$, where $\mathcal{M}^{unseen} \notin \mathcal{M}^{train}$. Consequently, the optimisation target shifts to $\eta = E_{\pi,\mathcal{M}^{unseen}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

This setting, which seeks to enhance the generalisability of agents in multiple distinct environments using offline datasets, is vital for future research in artificial general intelligence. It is important to note that the two primary variations between tasks $\mathcal{M}^k = \{S^k, A^k, r^k, P^k, \eta_0^k, \gamma\}$ are the differences in transition functions P^k and rewards r^k. Representation learning can be employed to unify S^k and A^k, and the distributions of η_0^k will be incorporated when modelling transition functions.

Although there has been some prior research addressing the multi-task problem in offline RL, it has predominantly been restricted to distinct tasks sharing the same dynamic function [26, 35]. Therefore, we emphasise multi-environment in this setting, specifically the differences in transition and reward functions. Multi-environment settings necessitate models with multiple generalisation capabilities, which has resulted in limited research in this area.

4 Task-prompt Generalised World Model

Recent studies have demonstrated the potential advantages of model-based RL approaches in offline settings. Furthermore, due to their flexibility in modifying reward functions and offering more stable updates, model-based RL methods excel in addressing multi-task problems. Therefore, it is logical to employ a model-based approach to tackle the multi-environment offline RL problem. This method incorporates a learned world model that can effectively serve as a surrogate for various purposes, including offline policy evaluation and learning, which requires the ability to generalise across different types of environments. Motivated by the success of multi-task prompt methods in vision and language domains [23], we opted to integrate this architectural inductive bias into the world model's learning process. Diverse environments may share identical mapping relationships, which can pose challenges in state prediction. Prompts can encode task-specific information, enabling sequence prediction models to disambiguate tasks effectively.

Therefore, we propose **TGW** framework, combining prompt and model-based method to solve the multi-environment offline RL problem. TGW is paradigm for learning a generalised world model $\hat{\mathcal{M}}^{global} = \{S, A, \hat{r}, \hat{P}, \hat{\eta}_0, \gamma\}$ from various datasets $\{(D^k)\}_{k=1}^K$. Even further, we assume that an environment $\mathcal{M}^k$ that represented a single task, and the state transfer space is a subset of generalised world model $\hat{\mathcal{M}}^{global}$. As mentioned above, TGW take into account the effect of distinct different transitions $\{(P^k)\}_{k=1}^K$ and rewards

$\{(r^k)\}_{k=1}^K$ on the final task. It is composed of three modules, a task-state prompter module, a generalised dynamics module, and a reward module. The general framework is shown in Figure 2. TGW has the same functions as the environment in which the agents interact, with current observation o_t and action a_t as input, current reward r_t and next observation o_{t+1} as output.

As policy optimisation relies on the world model, an accurate and dependable world model can generate the return under the policy $\eta_{\pi,\hat{\mathcal{M}}^{global}}$ such that it approximates as closely as possible the return in the exact environment $\eta_{\pi,\mathcal{M}^k}$. Therefore, the ultimate objective for TGW is to minimise the following expressions:

$$error = \sum_{k=1}^{K} \|\eta_{\pi,\hat{\mathcal{M}}^{global}} - \eta_{\pi,\mathcal{M}^k}\| \tag{1}$$

The **task-state prompter** extracts raw data features and subsequently integrates historical or task-related prompt information with current data to create a standardised prompt input for the next module. The first function, represented as $p(s_t|o_t)$, accomplishes the process of deducing the internal latent states s_t from local observations o_t at time step t. The second function can be formulated as $p(h_t|a_t, s_t, h_{t-1})$, where a_t and s_t represent the current action and state, respectively. We denote h_t as the task-state prompt at time step t. Furthermore, we presume that it contains potentially valid information about states and tasks, which effectively aids the subsequent module in accurately distinguishing between different tasks.

The **generalised dynamics module** constructs a unified dynamics model that incorporates each subset of task-specific environments. The input for this module is the task-state prompt h_t, and the outputs are the next state s_{t+1} and the next local observation o_{t+1}. Consequently, this module can be represented as $p(s_{t+1}, o_{t+1}|h_t)$.

Reward module constructs reward functions for different tasks. Reward function can be formulated as $p(r_t|h_t, s_{t+1}, o_{t+1})$.

In the following subsections, we will introduce TGW implementation and experimentally prove its validity.

5 Practical implementation

5.1 Trajectory Prompting

In the field of natural language processing, prompt-based methods have been employed to address zero-shot and few-shot challenges. Similarly, prompts have recently been demonstrated to provide effective task-specific instructions for disambiguating tasks in reinforcement learning policy optimisation [34, 22]. Distinct from these two works, TGW focuses on learning a generalised world model, rather than striving for optimal performance behaviour policy from offline datasets. Here, we utilise the sequential decision trajectory $(o_i, a_i, r_i, o_{i+1})_{i=1}^M$ from the offline data D^k as a task-state prompt h_t, where M represents the length of the trajectory contained in the prompt. We employ multi-layer perceptron neural networks (MLP) to project the original observation O onto the hidden space S and then tokenize serialised data. We define the task-state prompt h_t for task k at time t as follows:

$$h_t^k = (s_{t-m+1}^k, a_{t-m+1}^k, s_{t-m+2}^k, a_{t-m+2}^k, ..., s_t^k, a_t^k) \quad \text{where } s_i^k = f(o_i^k) \tag{2}$$

At the beginning of the evaluation, the tokens are derived from the task-specific dataset D^k. Subsequently, based on the first-in-first-out mechanism, the tokens are gradually replaced with historical information about the agent's interaction with the actual environment.

5.2 TransRSSM

To develop a dynamic model with enhanced generalisation and improved accuracy, we propose the TransRSSM network structure, which combines the advantages of the transformer and the recurrent state-space model (RSSM) [10]. The transformer is an apt architecture for modelling sequence prediction in a hidden space, owing to its ability to extract inter-relationships between contexts [32, 3]. In the RSSM, we assume that the latent space of dynamics consists of deterministic states s_t and stochastic states u_t, which have been empirically shown to be critical by PlaNet [10] and Dreamer [9]. Separating states into deterministic and stochastic categories bolsters the robustness of the world model predictions.

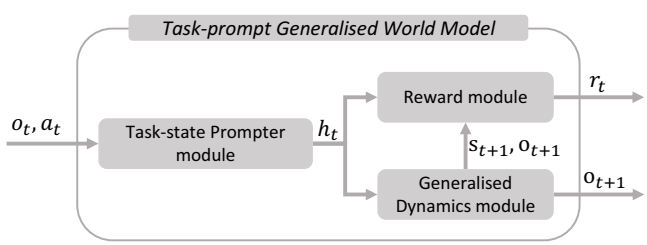

Figure 2. The TGW consists of three parts, the task-state prompter module, the generalised dynamics module, and the reward module.

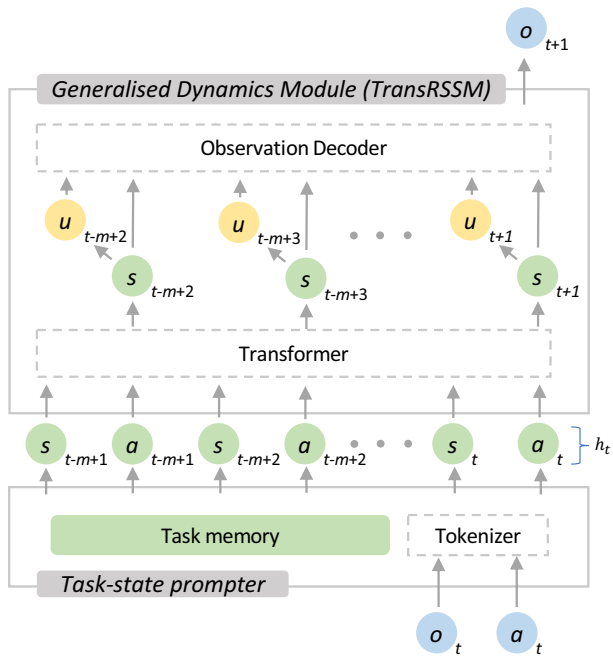

Figure 3. Task-state prompter and TransRSSM. In the diagram, the blue blocks represent data in the original space, the green blocks denote deterministic states in the latent space, and the yellow blocks correspond to stochastic states.

Algorithm 1 TransRSSM

Input: TransRSSM parameters θ_G, task-state prompt h_t
Output: Deterministic state s_{t+1}, observation o_{t+1}

TransRSSM parameters θ_G contains deterministic state model parameters θ_d, stochastic state model parameters θ_s, observation model parameters θ_o
Get deterministic state $s_{t+1} = f_{\theta_d}(h_t)$
Get means and variances of stochastic state $u_{t+1}^{mean}, u_{t+1}^{var} = f_{\theta_s}(s_{t+1})$
Sample stochastic state $u_{t+1} \sim \mathcal{N}(u_{t+1}^{mean}, u_{t+1}^{var})$
Get observation $o_{t+1} = f_{\theta_o}(s_{t+1}, u_{t+1})$
return deterministic state s_{t+1}, observation o_{t+1}

Figure 3 illustrates the token processing of task-state prompter and TransRSSM. The task-state prompter processes raw observations and actions at time t by converting them into embedded vectors, in which the raw observations are transformed into deterministic states. These vectors are then integrated with historical trajectories from the task memory to generate the serialised task-state prompt h_t. And h_t serves as the input for TransRSSM, supplying it with sufficient task-related historical information. The output of TransRSSM is the raw observations $o_{t+1} \in \mathcal{O}$ that will be sent to agent.

In TransRSSM, we initially utilise the transformer to predict the deterministic state s_{t+1} in the subsequent step. The generation of states becomes integrated with the contextual information encompassed in the task prompt h_t, capitalising on the transformer's sequence modelling capabilities. We split states into deterministic and non-deterministic states, which enables the model to make more accurate state predictions. In numerous experiments, this distinction has proven to be crucial [10, 25]. Assuming the stochastic states conform to a normal distribution, We predict the means and variances of the stochastic states, which can be expressed as $u_{t+1}^{mean}, u_{t+1}^{var} = f_{\theta_s}(s_{t+1})$. Stochastic states can be generated by sampling from this distribution: $u_{t+1} \sim \mathcal{N}(u_{t+1}^{mean}, u_{t+1}^{var})$. Ultimately, the subsequent observed state that can be received by the agent is predicted by the observation model $o_{t+1}^k = f(s_{t+1}^k, u_{t+1}^k)$.

Thus, our TransRSSM comprises the following components:

$$\begin{aligned} &\text{Deterministic state model:} && s_{t+1}^k = f(h_t^k) \\ &\text{Stochastic state model:} && u_{t+1}^k \sim p(u_{t+1}^k | s_{t+1}^k) \\ &\text{Observation model:} && o_{t+1}^k = f(s_{t+1}^k, u_{t+1}^k) \end{aligned} \tag{3}$$

The computational procedure of TransRSSM is illustrated in Algorithm 1.

5.3 Uncertainty estimation as intrinsic reward

The optimal approach to building a dynamics model involves obtaining sufficient data to cover the entire $S \times A \to S$ space. However, this is impractical, particularly for offline datasets that are inevitably subject to distribution drift. Consequently, we propose estimating the uncertainty of the generalised dynamics model in advance, which can be employed as an intrinsic reward for policy optimisation.

Estimating the uncertainty of functions is a significant area of research within the field of machine learning and has proven to be empirically valuable for world model learning in RL [7, 4]. We leverage the ensemble method to train N dynamics models with distinct initialization parameters $\{\theta_{G_n} | n \in [1:N]\}$. At the outset, these models map the same inputs into various predictions. As training iterations progress, these predictions converge to more consistent values, while the variance decreases. Ultimately, states not encompassed by the offline datasets exhibit high uncertainty, whereas sampled data display low unvertainty.

Algorithm 2 Task-prompt Generalised World Model

Input: Offline dataset D, number of task K, batch size B
Output: TGW model $\hat{\mathcal{M}}_{(f_{\theta_T}, f_{\theta_R}, f_{\theta_G})}$

Initialize Task-state Prompter parameters θ_T , Reward module parameters θ_R, Generalised Dynamics module θ_G
while not converged **do**
  for k=1 to K **do**
    for b=1 to B **do**
      Sample a trajectory τ_b^k from D^k
    end for
    Get a minibatch input $\tau^k = \{\tau_b^k\}_{b=1}^B$
    Get task-state prompt $h^k = f_{\theta_T}(\tau^k, D^k)$
  end for
  Mix task-state prompt $h = \{h^k\}_{k=1}^K$
  Get state and observation $s^{pred}, o^{pred} = f_{\theta_G}(h)$
  Get external reward $re^{pred} = f_{\theta_R}(h, s^{pred}, u^{pred})$
  Calculate loss $\mathcal{L} = \frac{1}{B}[(o - o^{pred})^2 + (re - re^{pred})^2]$
  $\theta_T \leftarrow \theta_T - \alpha \nabla_{\theta_T} \mathcal{L}$
  $\theta_G \leftarrow \theta_G - \alpha \nabla_{\theta_G} \mathcal{L}$
  $\theta_R \leftarrow \theta_R - \alpha \nabla_{\theta_R} \mathcal{L}$
end while
return TGW model $\hat{\mathcal{M}}_{(f_{\theta_T}, f_{\theta_R}, f_{\theta_G})}$

In offline RL, exploration in a uncertain state should typically be avoided. We quantify the model's uncertainty by measuring the variance of deterministic states predicted by various ensemble models, and utilise it as an intrinsic reward for policy training:

$$ri_{t+1}^k = Var(s_{t+1}^k) \tag{4}$$

Extrinsic rewards are typically represented by predicting a scalar based on state transitions and supervision labels. Additionally, the serialised latent states and the current deterministic and stochastic states are considered as inputs for extrinsic reward prediction, with the aim of aligning the reward prediction with task information and state-transition stochasticity. The mapping process can be expressed as follows:

$$re_{t+1}^k = f(h_t^k, s_{t+1}^k, u_{t+1}^k) \tag{5}$$

Here, we utilise β as a hyperparameter representing the level of risk tolerance of the agent, which balances exploration in uncertainty. The final reward expression is as follows:

$$r_t^k = re_{t+1}^k - \beta * ri_{t+1}^k, \tag{6}$$

where r_t^k serves as the immediate reward obtained during the interaction between the policy and the world model, guiding the agent to avoid the missing parts of the state space in the offline dataset.

5.4 Training

The model parameters are τ and its prediction probability distribution can be expressed as:

$$\log p_\tau(o_1, r, a_1, ..., o_M, a_M) = \sum_{m=1}^{M} \log p_\tau(o_m|o_1, a_1, ..., o_{m-1}, a_{m-1}) \tag{7}$$

As for model training supervision signals, there are labels corresponding to the prediction values o_m^k and r_m^k within the offline data. As our objective is to model the dynamics function, the loss of the behaviour policy is not computed for sequence tokens. Consequently, the training loss under k tasks can be expressed as follows:

$$\mathcal{L}(\theta) = -\sum_{k=1}^{K}\sum_{m=1}^{M}\Big(\log p_\theta(o_m^k|o_1^k, a_1^k, ..., o_{m-1}^k, a_{m-1}^k) + \log p_\theta(r_m^k|o_1^k, a_1^k, ..., o_{m-1}^k, a_{m-1}^k)\Big) \tag{8}$$

The algorithm process is shown in Algorithm 2. As our primary focus is on the learning of generalised world models, policy optimisation is not the priority of this research agenda. Both planning and learning methods are feasible for learning policies with our world model, and we ultimately chose to use the offline version of MBPO [11] to enable a fair comparison with the baseline.

6 Experiments

In our experiments, we concentrate on addressing the following questions concerning the implementation of our TGW framework:

(1). How does TGW perform on seen tasks?

(2). How does TGW perform on unseen tasks?

(3). How does each module of the TWG configuration affect the final outcome?

6.1 Experiments on seen tasks

In the first experiment, we opted to use benchmark tasks in offline RL, OpenAI gym's MuJoCo-based environments [30], to validate the generalisability of our model. We utilise offline data from D4RL [4], encompassing three environments (HalfCheetah, Hopper, and Walker2D) and four types of data (Random, Medium, Medium-Replay, and Medium-Expert). These tasks require the RL agent to learn to control various robot joints to accomplish the task. The characteristics of the four datasets are as follows: **Random**: the behaviour policy is randomly initialised. **Medium**: the behaviour policy is based on the SCA algorithm trained to a certain performance. **Medium-replay**: this data is obtained from the training behaviour policy to the medium level. **Medium-expert**: the behaviour policy consists of medium policy and expert policy. As the range of returns varies across scenarios, we normalise reward values for all tasks to range between 0 and 100, facilitating comparison between tasks.

The baseline algorithms we consider for comparison derive from two aspects. Firstly, we compare our algorithm with model-free offline methods, including CQL [14] and Prompt-DT [34], with Prompt-DT also utilising the prompt-transformer architecture as we do. Secondly, model-based offline reinforcement learning algorithms resemble our methods, and we opt to use MOPO [38] and MOReL [12] as baselines.

We assess the performance of algorithms in a multi-environment offline setting. Each algorithm is cross-trained using a mixture of data from different environments at the same level and subsequently evaluated individually within each environment. Consequently, there are four sets of experiments corresponding to four types of data, with the algorithms in each set tested across three environments. For a more comprehensive comparison, we also conducted empirical evaluations in single-environment scenarios, wherein each algorithm was trained solely with data pertinent to the test environment. To ensure a fair comparison, we enhance each algorithm by incorporating a similar tokenisation process, enabling adaptation to varying observation dimensions. As the observation dimension differs between tasks, we utilise a neural network to map it onto a hidden space with the same dimension as the token, which is equally applicable for actions.

The results, averaged over five random seeds, are displayed in Table 1. It can be observed that in the multi-environment setting, TGW achieves state-of-the-art (SOTA) results in 10 out of the 12 tests, particularly in random and medium-replay datasets. On average, TGW outperforms the other algorithms by a considerable extent. In contrast, the next best approach is Prompt-DT, which achieves SOTA results in the remaining 2 out of 12 tests. Both Prompt-DT and TGW methods with prompt exhibit greater adaptability to multi-task environments compared to other methods. In single-environment settings, although TGW is not specifically tailored for particular tasks, it still achieves comparable performance. TGW attains optimal performance in 4 out of the 12 tests, followed by Prompt-DT, which obtains optimal results in 3 out of the 12 tests. Generally, the performance of an algorithm declines when the setting changes from single-env to multi-env. This is due to the presence of issues in multi-task learning, such as catastrophic forgetting and task interference. However, it can be observed that TGW has the least decrease in scores, demonstrating more stable performance when facing tasks in various environments.

Furthermore, we discovered that model-based approaches outperform model-free methods within poor data quality, suggesting that poor behaviour policy has less impact on model-based algorithms. Overall, the model-based method, TGW, exhibits adaptability to diverse environments and excels in multi-environment offline reinforcement learning problems.

6.2 Experiments on unseen tasks

To address question (2), we examine two variants of widely-used simulated continuous control problems, including cheetah-vel and ant-dir [20]. In cheetah-vel, there are 40 tasks with different target velocities, while ant-direction comprises 50 tasks with varying goal directions. Our baseline for comparison is Prompt-DT, which exhibits strong performance in multi-task problems and can fine-tune to unseen tasks with small datasets.

Our algorithm leverages the same offline data as Prompt-DT. However, we train in a multi-environment setting. As in the previous experiment, we normalise the reward values for all tasks to a range between 0 and 100, allowing for comparison across different tasks. To investigate the impact of data distribution on multi-task generalisation performance, we selected ten tasks from each of the two environments at equal intervals. The first five tasks of each environment were used for training (ten in total), and the remaining five tasks were employed to evaluate performance on unseen tasks.

In prior model-based offline algorithms, generalisation to unseen tasks was achieved by directly assuming a known reward function, which effectively introduced task-related instructions for policy optimisation. However, in practice, it is frequently impossible to capture the exact reward function. As such, we evaluate how the reward module of TGW will perform under unseen tasks. To obtain a deeper understanding of the impact of the reward function on TGW, we experimentally set TGW to two versions: **TGW-finetune**, which fine-

Table 1. Results for multi-environment seen tasks. The table presents eight sets of experiments, corresponding to training configurations with four sets of single-environment data and four sets of multi-environment data. The results on the left side pertain to distinct models trained with single-environment datasets and evaluated within the respective environments. The results on the right side relate to a single model trained with multi-environment datasets and evaluated across the corresponding environments.

Test Environment	Prompt-DT	CQL	MOPO	MOReL	TGW(Ours)	Prompt-DT	CQL	MOPO	MOReL	TGW(Ours)
	Single-env Test 1: Training with random data					Multi-env Test 1: Training with random data				
halfCheetah	29.3	35.4	34.4	25.6	**36.2**	24.3	12.1	18.3	13.7	**29.7**
hopper	10.5	10.8	11.7	**53.6**	48.7	10.8	5.7	16.7	23.1	**37.2**
walker2d	18.6	7.0	13.6	37.3	**38.5**	17.6	3.4	14.2	12.6	**31.6**
	Single-env Test 2: Training with medium data					Multi-env Test 2: Training with medium data				
halfCheetah	**47.1**	44.4	42.3	42.1	46.1	**45.3**	18.1	27.8	12.4	41.2
hopper	55.3	86.6	28.0	95.4	**97.5**	58.9	49.3	27.1	28.7	**88.4**
walker2d	62.3	74.5	17.8	**77.8**	61.2	51.3	52.7	17.8	28.2	**60.3**
	Single-env Test 3: Training with medium-replay data					Multi-env Test 3: Training with medium-replay data				
halfCheetah	42.4	46.2	**53.1**	40.2	44.2	38.6	25.1	24.2	18.1	**40.5**
hopper	60.9	48.6	67.5	**93.6**	82.1	56.1	42.3	34.2	23.9	**70.6**
walker2d	47.1	32.6	39.0	49.8	**50.7**	43.2	25.4	18.2	29.7	**47.3**
	Single-env Test 4: Training with medium-expert data					Multi-env Test 4: Training with medium-expert data				
halfCheetah	**67.2**	62.4	63.3	53.3	59.3	**58.2**	46.4	42.6	28.2	51.1
hopper	98.3	**111**	23.7	108.7	103.2	92.4	83.2	31.7	42.7	**100.2**
walker2d	**103.7**	98.7	44.6	95.6	84.7	75.9	73.5	44.6	33.7	**86.7**
Average	53.3	54.9	32.8	**64.4**	62.7	47.7	36.4	26.5	24.6	**57.3**

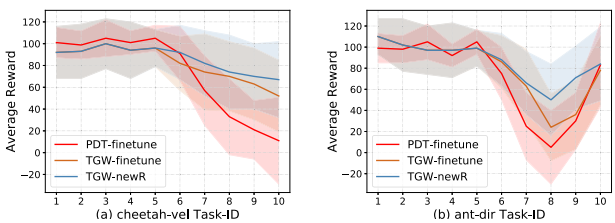

Figure 4. Results for multi-environment unseen tasks. Solid lines indicate means and shaded areas indicate variances. The five tasks on the left of both figures are seen tasks, while the others are unseen tasks.

tunes the reward function with the task-related dataset within the prompt, and **TGW-newR**, which trains policies directly using the reward function of the new task.

The experimental results, obtained over five random seeds, are depicted in Figure 4. As evident from the variations in the curves, both TGW and Prompt-DT exhibit strong performance on seen tasks. And their performance declines as the tasks deviate from the known environment distribution. In the cheetah-vel tasks, larger task IDs correspond to a greater degree of out-of-distribution task statuses. The performance of PDT-FineTune begins to degrade significantly, while our methods remain relatively stable. In contrast, the ant-dir tasks exhibit an opposite pattern, with Task 10 reverting to a distribution similar to that of Task 1. Consequently, the curve begins to rise at Tasks 9 and 10. In terms of overall results, TGW-NewR delivers the best performance, followed by TGW-FineTune.

6.3 Ablation study

To address question (3), we carry out a comprehensive ablation study on TGW. The primary objective is to investigate the impact of the three TGW modules on the final performance, corresponding to three algorithm configurations. **TGW w/o Prompt**: our model forgoes the task prompt and directly converts the current observation into a token, which is then input into the Generalised Dynamics module. **TGW w/o RSSM**: we cease sampling stochastic states. **TGW w/o ri**: we utilise only predicted external rewards in our policy optimisation. Regarding the experimental environment, we use the same configuration as in Experiment 1. The results of the experiment are presented in Table 2.

From the results, it can be seen that the complete TGW method performs the best in the majority of cases. Without the prompt, the performance noticeably declines in each test. We speculate that, upon removing the prompt, the transformer model's absence of context dependency information results in an accumulation of state prediction errors. The lack of adjustment of the length of the model's dependence on context led to irreparable deviations in policy optimization, which did not perform well even on single-env task.

The TransRSSM exerts a positive influence on learning in both single-environment and multi-environment settings, regardless of data quality. We believe that the reason for its effectiveness across various settings is due to the necessity of a world model, learned from offline data, that possesses the ability to handle random data distribution. This is because in offline RL, the dataset cannot cover all possible state transitions, resulting in epistemic uncertainty remaining, regardless of the presence or absence of aleatoric uncertainty. And the design of TransRSSM addresses environmental stochasticity, leading to more robust performance.

The final column presents ablation experiment results for intrinsic rewards, demonstrating a significant impact on the final outcomes in most cases. This can be attributed to the fact that, in the context of offline reinforcement learning, exploration in highly uncertain states may result in severe extrapolation errors, whereas intrinsic rewards encourage exploration in more conservative states. Additionally, the impact of intrinsic reward is related to the hyperparameter β. We investigated the effect of β, and the results of the ablation experiment over five random seeds are shown in Figure 5. It can be seen that the intrinsic rewards have a relatively smaller impact when the data qual-

Table 2. Results for ablation study. The table shows four sets of experiments, each corresponding to algorithms trained with different qualities of data

Test Environment	TGW	TGW w/o P	TGW w/o RSSM	TGW w/o ri	TGW	TGW w/o P	TGW w/o RSSM	TGW w/o ri
	Single-env Test 1: Training with random data				Multi-env Test 1: Training with random data			
halfCheetah	**36.2**	8.7	31.5	**36.2**	**29.7**	7.5	23.4	**29.7**
hopper	**48.7**	23.1	35.8	28.1	37.2	10.3	**40.5**	36.5
walker2d	**38.5**	18.5	20.3	31.4	**31.6**	16.6	24.9	28.0
	Single-env Test 2: Training with medium data				Multi-env Test 2: Training with medium data			
halfCheetah	**46.1**	40.7	27.6	33.8	**41.2**	22.0	31.5	**41.2**
hopper	**97.5**	37.9	54.1	77.0	**88.4**	33.7	28.4	72.3
walker2d	**61.2**	25.2	27.0	**61.2**	**60.3**	25.1	31.8	49.7
	Single-env Test 3: Training with medium-replay data				Multi-env Test 3: Training with medium-replay data			
halfCheetah	**44.2**	29.9	38.2	28.8	**40.5**	18.1	39.8	38.0
hopper	**82.1**	18.3	64.6	71.0	**70.6**	26.3	61.2	65.8
walker2d	**50.7**	39.5	33.2	**50.7**	**47.3**	19.0	34.9	34.4
	Single-env Test 4: Training with medium-expert data				Multi-env Test 4: Training with medium-expert data			
halfCheetah	**59.3**	30.0	39.8	42.7	**51.1**	12.7	36.7	42.3
hopper	**103.2**	7.5	70.2	91.6	**100.2**	28.0	78.3	65.2
walker2d	**84.7**	15.4	70.9	60.4	**84.2**	18.4	71.6	73.3

ity is poor, and a larger impact under medium-expert data. Although a trade-off exists, this configuration can indeed enhance the scope of policy exploration, and tuning of the hyperparameter β has an advantage in terms of improving the final result. In summary, through ablation experiments, we have validated that each module distinctly and positively influences the final performance of TGW.

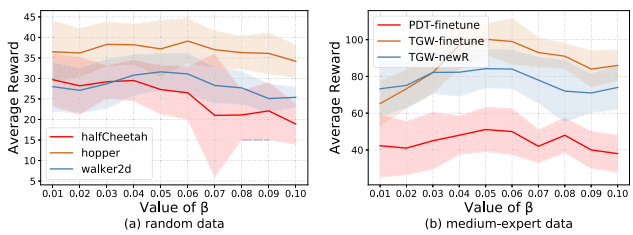

Figure 5. Ablation study of β. The two figures display the impact of β on under different data quality conditions.

7 Related work

To mitigate the substantial cost of online interaction and inefficient data utilisation, reinforcement learning has evolved from purely online RL to transition memory-based RL, commonly referred to as **offline RL** [15]. Due to extrapolation error [6], most offline RL algorithms constrain the policy to be within the domain of the behaviour policy. Some algorithms improve upon importance sampling [18, 28, 29], while others focus on quantifying the uncertainty of predictions [13] or training the policy with pessimistic Q-value estimation [6]. Additional works concentrate on offline data collection [36]. In contrast to these approaches, which rely on datasets to train specialised policies, TGW extracts generic information from data across different environments.

Model-Based RL has demonstrated remarkable success in complex decision-making problems, with planning methods to predict further horizons [5, 27, 24]. Compared to model-free methods, Model-Based RL offers advantages in stability, data efficiency, and security [21, 8]. There are methods that use world models to facilitate transfer across tasks with different reward functions to accomplish multiple tasks [25]. CASCADE [35] learns a generalised world model in an online setting. Some methods are based on offline data. ALPT is based on data pre-training for inverse dynamics models [33]. And MOReL [12], MOPO [38], and COMBO [37] learn environment models with uncertainty-based punishment on offline data. However, these world models remain restricted to a single dynamic environment. As a generalised world model, TGW can adapt to different tasks with both diverse reward functions and distinct transition functions.

Transformer-based RL methods often models state-action trajectories as a sequence prediction problem, simplifying a range of design decisions, as seen in the decision transformer [3, 22]. However, these methods conflate world models with decision-making, rendering planning infeasible in these algorithms. Prompt transformers were initially used to address problems without large amounts of annotated data [19] and have proven useful for tackling tasks with data from various distributions [23]. Both GATO [22] and Prompt-DT [34] employ prompt transformers to solve RL problems and are adaptable to multiple tasks. The distinction lies in TGW's focus on constructing a generalised model rather than directly imitating policy. The advantage of our method is that policy learning is not constrained by behaviour policy.

8 Conclusions

We propose a novel multi-environment offline RL setting in which the agent leverages fixed demonstrations collected from different environments to perform well across distinct tasks. To address this, we introduce a framework called TGW that incorporates the architectural inductive bias of the prompt-transformer. During the algorithm's implementation, we consider the properties of constructing a world model in RL and propose the structure named TransRSSM. Experimental results demonstrate TGW's superior performance for both seen and unseen tasks compared to previous offline RL algorithms. We also confirm that the task-state prompter and TransTSSM are critical to the final outcome.

For future work, our main direction is to extend its generalization across different dynamics and expand towards multi-modal states.

9 Acknowledgement

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant (No.XDA27030300), and the Program for National Nature Science Foundation of China (62073324).

References

[1] Christopher G Atkeson and Juan Carlos Santamaria, 'A comparison of direct and model-based reinforcement learning', in *Proceedings of international conference on robotics and automation*, volume 4, pp. 3557–3564. IEEE, (1997).

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch, 'Decision transformer: Reinforcement learning via sequence modeling', *Advances in neural information processing systems*, **34**, 15084–15097, (2021).

[4] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine, 'Deep reinforcement learning in a handful of trials using probabilistic dynamics models', *Advances in neural information processing systems*, **31**, (2018).

[5] Andreas Draeger, Sebastian Engell, and Horst Ranke, 'Model predictive control using neural networks', *IEEE Control Systems Magazine*, **15**(5), 61–66, (1995).

[6] Scott Fujimoto, David Meger, and Doina Precup, 'Off-policy deep reinforcement learning without exploration', in *International conference on machine learning*, pp. 2052–2062. PMLR, (2019).

[7] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al., 'A survey of uncertainty in deep neural networks', *arXiv preprint arXiv:2107.03342*, (2021).

[8] David Ha and Jürgen Schmidhuber, 'World models', *arXiv preprint arXiv:1803.10122*, (2018).

[9] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi, 'Dream to control: Learning behaviors by latent imagination', *arXiv preprint arXiv:1912.01603*, (2019).

[10] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson, 'Learning latent dynamics for planning from pixels', in *International conference on machine learning*, pp. 2555–2565. PMLR, (2019).

[11] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine, 'When to trust your model: Model-based policy optimization', *Advances in Neural Information Processing Systems*, **32**, (2019).

[12] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims, 'Morel: Model-based offline reinforcement learning', *Advances in neural information processing systems*, **33**, 21810–21823, (2020).

[13] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine, 'Stabilizing off-policy q-learning via bootstrapping error reduction', *Advances in Neural Information Processing Systems*, **32**, (2019).

[14] 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).

[15] Sascha Lange, Thomas Gabel, and Martin Riedmiller, 'Batch reinforcement learning', in *Reinforcement learning*, 45–73, Springer, (2012).

[16] Yann LeCun, 'A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27', (2022).

[17] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu, 'Offline reinforcement learning: Tutorial, review, and perspectives on open problems', *arXiv preprint arXiv:2005.01643*, (2020).

[18] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang, 'Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms', in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 297–306, (2011).

[19] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig, 'Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing', *arXiv preprint arXiv:2107.13586*, (2021).

[20] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn, 'Offline meta-reinforcement learning with advantage weighting', in *International Conference on Machine Learning*, pp. 7780–7791. PMLR, (2021).

[21] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker, 'Model-based reinforcement learning: A survey', *arXiv preprint arXiv:2006.16712*, (2020).

[22] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al., 'A generalist agent', *arXiv preprint arXiv:2205.06175*, (2022).

[23] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al., 'Multitask prompted training enables zero-shot task generalization', *arXiv preprint arXiv:2110.08207*, (2021).

[24] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al., 'Mastering atari, go, chess and shogi by planning with a learned model', *Nature*, **588**(7839), 604–609, (2020).

[25] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak, 'Planning to explore via self-supervised world models', in *International Conference on Machine Learning*, pp. 8583–8592. PMLR, (2020).

[26] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman, 'Dynamics-aware unsupervised discovery of skills', *arXiv preprint arXiv:1907.01657*, (2019).

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., 'Mastering chess and shogi by self-play with a general reinforcement learning algorithm', *arXiv preprint arXiv:1712.01815*, (2017).

[28] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade, 'Learning from logged implicit exploration data', *Advances in neural information processing systems*, **23**, (2010).

[29] Adith Swaminathan and Thorsten Joachims, 'Batch learning from logged bandit feedback through counterfactual risk minimization', *The Journal of Machine Learning Research*, **16**(1), 1731–1755, (2015).

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa, 'Mujoco: A physics engine for model-based control', in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, (2012).

[31] Harm Van Seijen, Hadi Nekoei, Evan Racah, and Sarath Chandar, 'The loca regret: a consistent metric to evaluate model-based behavior in reinforcement learning', *Advances in Neural Information Processing Systems*, **33**, 6562–6572, (2020).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, 'Attention is all you need', *Advances in neural information processing systems*, **30**, (2017).

[33] David Venuto, Sherry Yang, Pieter Abbeel, Doina Precup, Igor Mordatch, and Ofir Nachum, 'Multi-environment pretraining enables transfer to action limited datasets', *arXiv preprint arXiv:2211.13337*, (2022).

[34] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan, 'Prompting decision transformer for few-shot policy generalization', in *International Conference on Machine Learning*, pp. 24631–24645. PMLR, (2022).

[35] Yingchen Xu, Jack Parker-Holder, Aldo Pacchiano, Philip J Ball, Oleh Rybkin, Stephen J Roberts, Tim Rocktäschel, and Edward Grefenstette, 'Learning general world models in a handful of reward-free deployments', *arXiv preprint arXiv:2210.12719*, (2022).

[36] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto, 'Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning', *arXiv preprint arXiv:2201.13425*, (2022).

[37] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn, 'Combo: Conservative offline model-based policy optimization', *Advances in neural information processing systems*, **34**, 28954–28967, (2021).

[38] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma, 'Mopo: Model-based offline policy optimization', *Advances in Neural Information Processing Systems*, **33**, 14129–14142, (2020).