Pre-trained Language Models Improve the Few-shot Prompt Ability of Decision Transformer

Yu Yang
Duke University
yu.yang@duke.edu

Pan Xu
Duke University
pan.xu@duke.edu

Abstract

Decision Transformer (DT) has emerged as a promising class of algorithms in offline reinforcement learning (RL) tasks, leveraging pre-collected datasets and Transformer's capability to model long sequences. Recent works have demonstrated that using parts of trajectories from training tasks as prompts in DT enhances its performance on unseen tasks, giving rise to Prompt-DT methods. However, collecting data from specific environments can be both costly and unsafe in many scenarios, leading to suboptimal performance and limited few-shot prompt abilities due to the data-hungry nature of Transformer-based models. Additionally, the limited datasets used in pre-training make it challenging for Prompt-DT type of methods to distinguish between various RL tasks through prompts alone. To address these challenges, we introduce the Language model-initialized Prompt Decision Transformer (LPDT), which leverages pre-trained language models for meta-RL tasks and fine-tunes the model using Low-rank Adaptation (LoRA). We further incorporate prompt regularization to effectively differentiate between tasks based on prompt feature representations. Our approach integrates pre-trained language model and RL tasks seamlessly. Extensive empirical studies demonstrate that initializing with a pre-trained language model significantly enhances the performance of Prompt-DT on unseen tasks compared to baseline methods.

1 Introduction

In many sequential decision-making applications such as robotic manipulation and autonomous driving [28, 15], it can be expensive or even unsafe for agents to learn through trial-and-error with the environment. Offline reinforcement learning (RL) methods [18] have emerged as a powerful paradigm for optimizing agent policies without directly interacting with the environment. They leverage pre-collected datasets obtained from a set of behavior policies instead of online interactions to learn an optimal policy. Among these Offline RL methods, Decision Transformer (DT) [3] has become popular for offline RL tasks due to its scalability with computation and data and stability in training. DT models a goal-conditioned policy using a Transformer network, solving a sequence-prediction problem in a supervised learning manner. Compared with traditional dynamic programming-based offline RL methods [13, 6, 14] that heavily rely on the Markov Decision Process (MDP) assumption of the environment, DT can utilize entire trajectory histories to predict the next action, making them more applicable in partially observable environments where all past information must be incorporated in decision-making[12, 21]. In addition to the context ability, another advantage of Transformers is their few-shot generalization ability [2, 1]. Based on their remarkable few-shot generalization capability, a prompt-based framework has been proposed and proven effective for adapting to new tasks in NLP [2, 20]. In this paradigm, the prompt, containing useful information about the task, is inputted as a prefix to the model for identifying the environments. Previous works [31] have demonstrated that DTs also exhibit good generalization ability for unseen tasks.

NeurIPS 2024 Workshop on Adaptive Foundation Models.

However, existing Prompt-DT methods [31, 9, 10] require significant amounts of data for pre-training due to the data-hungry nature of Transformers [2, 1]. Offline RL datasets are often small and insufficient to fully unleash the few-shot prompt learning capability of Transformers. Collecting large amounts of RL trajectories for pre-training powerful Decision Transformers is challenging. Inspired by the broad success of large language models in NLP, recent works [19, 25, 27] have shown the potential of such models to provide effective initial weights for decision-making tasks. However, these works do not directly demonstrate few-shot abilities due to a lack of multi-task training and prompt guidance. Language initialization in these works provides pre-knowledge and helps alleviate the need of huge datasets. Therefore, we aim to explore the use of pre-trained language models to initialize Prompt-DT methods and reduce the dependency on large datasets for training.

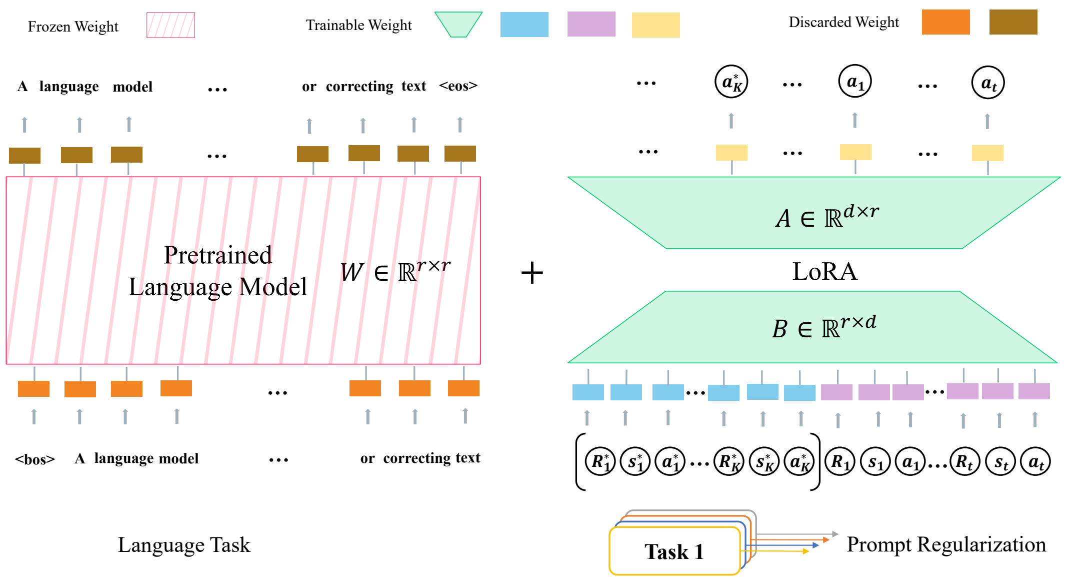

Frozen Weight
Trainable Weight
Discarded Weight
A language model ... or correcting text <eos>
Pretrained Language Model
$W \in \mathbb{R}^{r\times r}$
+
$A \in \mathbb{R}^{d\times r}$
LoRA
$B \in \mathbb{R}^{r\times d}$
<bos> A language model ... or correcting text
R_1^* s_1^* a_1^* ... R_K^* s_K^* a_K^* R_1 s_1 a_1 ... R_t s_t a_t
... a_K^* ... a_1 ... a_t
Language Task
Task 1
Prompt Regularization

Figure 1: Overview of LPDT. We first initialize our algorithm using a pre-trained language model such as DistilGPT2 [26]. Our method LPDT replaces the word embedding layers with linear layers to fully learn and capture the features of RL trajectory tokens. We fine-tune our model using parameter-efficient methods like Low-Rank Adaptation (LoRA). Additionally, we incorporate prompt regularization over the input prompts , which helps LPDT distinguish between different environments.

In this work, we propose a novel framework, Language model-initialized Prompt Decision Transformer (LPDT), that utilizes pre-trained language model initialization to improve the few-shot prompt ability of Decision Transformer. Our approach initializes the model with pre-trained language models, incorporating pre-existing knowledge that might benefit downstream RL tasks. A more detailed illustration of the model structure and our training paradigm is provided in Figure 1. We conduct extensive experiments to assess the capability of our proposed framework in MuJoCo control environments [4] and Meta World ML1 tasks [34]. Our method outperforms baselines in terms of cumulative rewards on unseen tasks. Our contributions are summarized as follows.

- We propose a framework named LPDT to improve the few-shot prompt ability of Decision Transformer. This framework involves leveraging the language model as the initialization of DT and imposing both supervised and unsupervised prompt regularization. LPDT demonstrates improved few-shot prompt capabilities with pre-trained language knowledge in multi-task RL.
- We utilize Low-Rank Adaptation (LoRA) and an additional prompt regularization method to combine pre-trained knowledge with domain-specific RL task knowledge. LoRA allows efficient fine-tuning by adapting only a small subset of parameters, while both the supervised and unsupervised prompt regularization enhances the model's ability to distinguish task information contained in the prompts.
- Through extensive experiments on MuJoCo control and Meta World ML1, we demonstrate the advantages of LPDT compared to baselines. Our results show that LPDT significantly outperforms existing models in performance under full and limited datasets, highlighting its potential for real-world applications.

2 Preliminary

2.1 Offline Reinforcement Learning

Reinforcement learning problems are usually formulated as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, d_0, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the set of states $s \in \mathcal{S}$, $\mathcal{A}$ represents the set of actions $a \in \mathcal{A}$, $\mathcal{T}$ is the transition distribution in the form $\mathcal{T}(s_{t+1}|s_t, a_t)$, d_0 is the distribution of initial states s_0, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $r_t = \mathcal{R}(s_t, a_t)$ is the reward at timestep t, and $\gamma \in (0, 1)$ is the discount factor. The objective is to find a policy π that maximizes the expected cumulative rewards $J(\pi)$:

$$J(\pi) = \mathbb{E}_{s_0 \sim d_0(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)} \left[\textstyle\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right]. \tag{2.1}$$

Among various offline RL methods [18, 13, 6, 14], Decision Transformer [3] which leverages the Transformer [29] to predict the next action conditioned on the past trajectory is drawing increasing attention. In DT, the trajectories $\{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T\}$ in the offline dataset $\mathcal{D}$ are reformulated and modeled as a sequence generation problem via self-supervised learning paradigm.

2.2 Prompt Decision Transformer

Xu et al. [31] introduced Prompt Decision Transformer, which leverages the DT architecture to model the RL trajectories in multi-task environments and make decisions in unseen tasks. In the offline dataset D, we have the trajectories τ. The rewards in the training trajectories of DT are replaced by the return-to-go denoted as $R_i = \sum_{t=i}^{T} r_t$. The prompt is the short trajectory from the dataset. During the training stage, we utilize the offline RL dataset $\mathcal{D}$ which contains multiple RL tasks denoted as $T_i \in T^{train}$. The input of Prompt-DT is a concatenation of the prompt and the training trajectory, denoted by τ_i^* and τ_i respectively. We have the input vector τ_i^{input} defined as

$$\tau_i^{input} = [\tau_i^*, \tau_i] = \left(R_{i,1}^*, s_{i,1}^*, a_{i,1}^*, \cdots, R_{i,K^*}^*, s_{i,K^*}^*, a_{i,K^*}^*, R_{i,1}, s_{i,1}, a_{i,1}, \ldots, R_{i,K}, s_{i,K}, a_{i,K}\right). \tag{2.2}$$

where K^* is the length of the prompt and K is the length of the training trajectories. Besides, we denote the partial trajectory from the timestep 1 to timestep t as $\tau_{i,1<t}^{input}$. Then the learning objective of Prompt-DT can be formulated as the following maximum likelihood estimation:

$$L_{PDT} = \mathbb{E}_{\tau_i^{input} \sim T_i} \left[\textstyle\sum_{t=1}^{K} - \log M_\theta(\hat{a}_{i,t} | \tau_i^*, \tau_{i,1<t-1}^{input}, R_{i,t}, s_{i,t}) \right]. \tag{2.3}$$

where M_θ denotes the Prompt-DT model with the parameter θ. In practical implementations, we often use the mean squared error loss instead, which aims to predict the future action $\hat{a}_{i,t}$ given the history trajectory and current state by minimizing the following loss function.

$$L_{PDT} = \mathbb{E}_{\tau_i^{input} \sim T_i} \left[1/K \textstyle\sum_{t=1}^{K} (a_{i,t} - \hat{a}_{i,t})^2 \right]. \tag{2.4}$$

The training procedure of Prompt-DT is to autoregressively generate the action conditioned on the current state, return-to-go, past trajectory and sampled prompt.

3 The Proposed Framework

We propose Language model-initialized Prompt Decision Transformer (LPDT), a novel and effective framework to incorporate powerful pre-trained language models into Decision Transformers to improve their few-shot learning abilities. We also leverage an additional prompt regularization over the prompts during fine-tuning to better identify tasks. Figure 1 illustrates the overview of our method. Algorithm 1 in the Appendix B demonstrates more details of this approach.

3.1 Language model initialization for Prompt-DT

The first step in our LPDT framework is to use pre-trained language models as the initialization. In this work, we use the DistilGPT2 [26, 24] as the initial weights, which is a pre-trained model with 82 million parameters and is faster and lighter than the original GPT-2 [24]. The common next-token prediction objective of GPTs can be formulated as

$$L_{LM} = \textstyle\sum_{i=1}^{t-1} - \log(M_{\theta^*}(w_{i+1} \mid w_1, \ldots, w_i)), \tag{3.1}$$

where M_{θ^*} is the language model and w_i represents the word token. To make the language model compatible with the RL sequence prediction tasks in DT, we follow previous work [27] to replace the word token embedding input and output layers with linear layers, which are trainable for specific RL tasks.

3.2 Parameter-efficient fine-tuning on RL tasks

To adapt the language models to specific RL tasks, we add a low-rank adaptation of the frozen weights of the language model and update it using parameter-efficient methods like LoRA [8]. Specifically, LoRA utilizes two low-rank matrices to represent the weight matrix, significantly reducing the number of parameters. This process can be formulated as $W = W_0 + \Delta W = W_0 + AB$, where $W \in \mathbb{R}^{d\times k}$ is the weight of our model, $W_0 \in \mathbb{R}^{d\times k}$ is the frozen weight inherited from the language model, and $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ are low-rank matrices. In this way, we avoid fully fine-tuning the language model and make our method scalable to large language models, where only a small number of parameters from the low-rank matrix ΔW need to be updated.

3.3 Prompt regularization with supervised and unsupervised objectives

Previous works [9, 10] aim to tune the prompt during testing on unseen tasks. However, they are not always effective when tasks are too similar. To address this challenge and achieve improved performance on testing tasks, we incorporate an additional regularization on the training loss over the prompt.

Since our model is built upon Prompt-DT, we use the loss function for Prompt-DT defined in (2.4) as the base loss function, and then incorporate a prompt regularization term. The loss function of our method is as follows.

$$L_{\text{total}} = \mathbb{E}_{\tau_i^{input} \sim T_i} \left[1/K \sum_{t=1}^{K} (a_{i,t} - \hat{a}_{i,t})^2 \right] + \lambda L_\phi, \tag{3.2}$$

where L_ϕ is the loss for the prompt regularization which we will specify in the rest of this section, and λ is the hyperparameter for prompt regularization.

Supervised learning-based prompt regularization. In this approach, we add a classifier head to the output of the prompt encoder. We use the task ID from the dataset as the label We adopt cross-entropy as the loss function. We formulate L_ϕ as:

$$L_\phi^{\text{classifier}} = -\sum_i y_i \log(\hat{y}_i), \tag{3.3}$$

where y_i is the true task label which means the task ID and $\hat{y}_i$ is the predicted probability for the task which comes from the prompt τ_i^*.

Unsupervised learning-based prompt regularization. When task IDs are unknown, the supervised method may not be feasible. Therefore, we also propose an unsupervised learning method We use the InfoNCE objective [22] to calculate the loss over the prompt. We formulate L_ϕ as:

$$L_\phi^{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{N} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \right], \tag{3.4}$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are the encoded representations of the prompts τ_i^* and τ_j^* respectively. The term $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$ denotes the similarity function (e.g., cosine similarity) between $\mathbf{z}_i$ and $\mathbf{z}_j$.

4 Experiments

In this section, we conduct experiments to evaluate the few-shot generalization ability of our proposed method LPDT. We evaluate the performance of LPDT on MuJoCo control tasks [4] and Meta World [34] with the episode accumulated reward as the evaluation metric. We also evaluate the prompt ability of LPDT with the smaller dataset sizes. More implementation details can be found in Appendix D.

4.1 Datasets and Tasks

In this work, we evaluate the performance of our proposed approach on MuJoCo controls and Meta World, which are commonly used in existing Prompt-DT type of methods [31, 9, 10], namely, Cheetah-dir, Cheetah-vel, Ant-dir, Meta-World reach-v2 and MW pick-place-v2. More details can refer to the Appendix C.1.

We compare the few-shot generalization ability of our proposed LPDT with baseline algorithms. For each method, we compare the performance based on the accumulated reward. The baselines we choose include Prompt-DT [31], Prompt-Tuning DT [9], and Prompt Diffuser [10]. Details of the baselines can be found in Appendix C.2.

4.2 Comparison with Prompt-DT type of methods

Table 1: Results for MuJoCo control tasks and MW tasks. The best mean scores are highlighted in bold. For each environment, the length of the prompt is $K^* = 5$. The dataset we utilized is the full dataset. We test all the results on unseen tasks with three random seeds. LPDT outperforms baselines on the Cheetah environment and is competitive in the Ant environment.

Task	Prompt-DT	Prompt-Tuning DT	Prompt Diffuser	LPDT-Classifier	LPDT-InfoNCE
Cheetah-dir	933.91±7.04	941.5±3.2	945.3±7.2	947.84±1.53	**951.72**±**4.08**
Cheetah-vel	-34.71±2.80	-40.1±3.8	-35.3±2.4	**-31.57**±**2.70**	-35.98±7.15
Ant-dir	396.07±9.78	427.9±4.3	**432.1**±**6.7**	374.13±23.05	412.47±21.01
MW reach-v2	**692.29**±**9.32**	472.5±29.0	555.7±6.8	497.61 ± 48.15	528.21±114.24
MW pick-place-v2	**3773.82**± **356.05**	-	-	3508.12±243.93	3543.38±191.32

We conduct experiments on our proposed LPDT and baseline methods to evaluate their performance. Table 1 demonstrates that our LPDT outperforms the baseline algorithms on MuJoCo Control tasks but suffers from inferior performance in Meta World compared with Prompt-DT. The possible limitation may be due to the huge difference between the RL task and the language task. Table 1 also illustrates that our approach performs better than the baselines in Cheetah-dir and Cheetah-vel, while it is not as good as Prompt Diffuser in Ant-dir but still better than Prompt-DT. For the results in the Meta World task, we report the results of Prompt-Tuning DT and Prompt Diffuser from their papers. We then conduct an ablation study on data efficiency. We split the dataset into different ratios and train algorithms on these split datasets. Table 2 demonstrates that LPDT outperforms the baseline method Prompt DT in different ratios which shows that the pre-trained language model incorporates prior knowledge about these RL downstream tasks.

Table 2: Ratio results for MuJoCo control tasks different ratio dataset. The best mean scores are highlighted in bold. For each environment, the length of the prompt is $K^* = 5$. We test all the results on unseen tasks with three random seeds. We demonstrate that language initialization provides prior knowledge for training on RL tasks and performs better than Prompt-DT with less data.

Tasks	Methods	Full	0.5	0.1	0.05	0.01
Cheetah-dir	Prompt DT	933.91±7.04	935.17±11.56	890.35±11.95	838.33±9.14	475.13±14.17
	LPDT-Classifier	947.84±1.53	**940.41**±**12.68**	**931.42**±**6.40**	911.5768±7.35	**806.82**±**10.34**
	LPDT-InfoNCE	**951.72**±**4.08**	940.27±4.18	919.75±4.17	**914.28**±**4.90**	774.33±21.62
Cheetah-vel	Prompt DT	-34.71±2.80	-41.08±8.99	-42.46±9.25	-45.38±4.30	-69.78±0.74
	LPDT-Classifier	**-31.57**±**2.70**	-37.45±4.01	-34.37±8.58	**-42.83**±**2.58**	**-65.61**±**6.14**
	LPDT-InfoNCE	-35.98±7.15	**-35.75**±**2.30**	**-34.20**±**8.21**	-44.1793±1.52	-78.32±13.64
Ant-dir	Prompt DT	396.07±9.78	**411.205**±**29.87**	361.13±4.46	**397.2984**±**22.93**	334.7232±6.65
	LPDT-Classifier	374.13±23.05	394.51±23.88	347.12±28.48	356.7666±22.79	**349.91**±**19.29**
	LPDT-InfoNCE	**412.47**±**21.01**	392.29±37.54	**369.73**±**21.97**	336.6543±39.43	314.34±4.96

5 Conclusion

In this work, we proposed a novel framework for improving the few-shot prompt ability of decision transformers in offline reinforcement learning, i.e., Language model-initialized Prompt Decision Transformer (LPDT). By leveraging pre-trained language models and combining them with domain-specific RL datasets, LPDT demonstrates improved few-shot prompt capabilities and outperforms or is competitive with existing baselines in prompt-based methods in terms of cumulative rewards on unseen tasks. Our approach has the potential to significantly reduce the data requirements for offline RL tasks, making it more applicable to real-world scenarios where collecting large amounts of RL trajectories is challenging. Furthermore, our results demonstrate the effectiveness of our prompt regularization methods in enhancing the model's ability to distinguish task information contained in prompts.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[5] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

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

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

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[9] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt-tuning decision transformer with preference ranking. *arXiv preprint arXiv:2305.09648*, 2023.

[10] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt tuning with diffusion for few-shot pre-trained policy generalization, 2024. URL `https://openreview.net/forum?id=7rex8lEZH2`.

[11] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[12] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[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] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

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

[17] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[18] 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.

[19] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[20] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[21] Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[23] Raul Puri and Bryan Catanzaro. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*, 2019.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[25] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

[26] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[27] Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon S Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. *arXiv preprint arXiv:2310.20587*, 2023.

[28] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pages 907–917. PMLR, 2022.

[29] 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.

[30] Zhihui Xie, Zichuan Lin, Deheng Ye, Qiang Fu, Yang Wei, and Shuai Li. Future-conditioned unsupervised pretraining for decision transformer. In *International Conference on Machine Learning*, pages 38187–38203. PMLR, 2023.

[31] 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*, pages 24631–24645. PMLR, 2022.

[32] Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. *arXiv preprint arXiv:2304.08487*, 2023.

[33] Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=AatUEvC-Wjv`.

[34] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[35] Xiangyuan Zhang, Weichao Mao, Haoran Qiu, and Tamer Başar. Decision transformer as a foundation model for partially observable continuous control. *arXiv preprint arXiv:2404.02407*, 2024.

A Related Work

Decision Transformer. Decision Transformer (DT) [3] emerged as a type of algorithm for offline RL by using the powerful Transformer architecture for decision-making. DT models RL trajectories as a sequence generation problem and utilizes the next-token generation paradigm for training. Thus, DT takes the history tokens such as the return, state, and action to predict the next action, which formulates the decision-making as an action prediction or sequence generation in a supervised fashion. Since DT can fully utilize the whole trajectories and is easy to train compared with dynamic programming-based offline RL, many of the following works improved the performance under different settings. For example, Lee et al. [16] proposed the Multi-game Decision Transformer which is trained on part of the Atari games as the multi-tasks training and fine-tuned on the remaining games to achieve efficient adaption. Hyper Decision Transformer [32] adds an adapter into Decision Transformer and is fine-tuned on unseen tasks through the demonstration without expert actions. Xie et al. [30] proposed to predict the action conditioned on the future trajectory embedding instead of conditioned on the return. Trajectory Transformer [11] is another research line, which is trained on sequences of state, action, and rewards and generated with the beam search.

Prompt-DT. Prompt Decision Transformer [31] utilizes the prompt-based framework to do the meta-RL. It is trained on multi-RL tasks with offline datasets. During the training, the prompts or demonstrations which are a small part of the trajectory are combined with trajectories. During the testing on unseen tasks, the prompt can be a guide for indicating the tasks and help the model predict the action to interact with the environments. Following Prompt-DT, several works are adopting the prompt tuning method to achieve a high-quality prompt. Prompt-Tuning DT [8] uses the preference ranking function and black-box tuning method to tune the prompt when testing on unseen tasks to achieve a high-quality prompt. Moreover, Prompt Diffuser [10] leverages the diffusion model to generate high-quality prompts leading to improved performance in downstream RL tasks. Different from these works, we adopt the prompt regularization which aims to learn a high-quality prompt embedding to distinguish the different but similar RL tasks. Our method adopts this regularization during the training procedure in the prompt dataset.

Language model based DT. Large language models have achieved many surprising effects in various tasks in recent years. Pre-trained on large datasets such as the corpus of the Internet, LLMs such as GPTs [24] demonstrate prompt ability which can generate the text with the guide of the task information. The success of the large language models motivates the increasing use of pre-trained language models in improving Decision Transformer to solve RL tasks [3]. Several works utilize the powerful representation generalization ability of language models as policies to do the decision-making. Li et al. [19] proposed to adopt the pre-trained language models for interactive decision-making to convert the policies to sequence data. Wik-RL [25] uses a pre-trained language model from the next-token generation paradigm as the initialization of DT for offline RL tasks. However, it suffers from inferior performance than directly using DT. To overcome these challenges and unleash the power of language models, Shi et al. [27] proposed the LaMo algorithm which uses a pre-trained language model and parameter-efficient fine-tuning methods to improve the original DT. Zhang et al. [35] also proposed to use LaMo in partially observable continuous control problems which demonstrates a strong generalization ability. Unlike all the above methods, our approach is fine-tuned for learning to identify different prompts for various RL tasks. And during the testing phase, just a small part of the trajectories is used in our method as the prompt without updating the model.

B Proposed Algorithm

C Environment and Baselines

C.1 Experiment Environments

We evaluate our approach over the MuJoCo tasks and Meta-World ML1 tasks. In Cheetah-dir, there are two tasks with goal directions as forward and backward, where the reward function promotes high velocity along the goal direction. The training and testing phases both include the two tasks. Similar to Cheetah-dir, Ant-dir also segments the tasks by directions. There are 50 tasks in Ant-dir with different goal directions uniformly sampled in 2D space. The tasks are split into 45 training tasks and

Algorithm 1: Language model-initialized Prompt Decision Transformer (LPDT)

```
Input: Pre-trained language model weights $\theta^*$, training datasets $\mathcal{D}$ with prompts and trajectories,
       prompt regularization hyperparameter $\lambda$
Output: Language Initialized Prompt Decision Transformer
1  Initialize Decision Transformer with pre-trained language model weights (e.g., DistillGPT2);
2  Replace the input and output embedding layers of the language model with linear layers;
3  for each epoch do
4      for each batch in training dataset do
5          Extract prompts $\tau_i^*$ and trajectories $\tau_i$ from the batch;
6          Encode prompts $\tau_i^*$ using the prompt encoder $\phi(\tau_i^*)$;
7          Transform trajectories $\tau_i$ using DT with prompt embeddings $\phi(\tau_i^*)$;
8          Compute the Prompt-DT loss $L_{\text{PDT}}$ over the input trajectories $\tau_i^{input}$ by (2.4); Compute
             the prompt regularization loss $L_\phi$ using the supervised learning (3.3) or unsupervised
             learning loss (3.4);
9          Combine the Prompt-DT loss and prompt regularization loss: $L_{\text{total}} = L_{\text{PDT}} + \lambda L_\phi$;
10         Backpropagate the combined loss and update parameters using parameter-efficient
             methods like LoRA;
11         Freeze the remaining weights and update only low-rank matrices $A$ and $B$:
                 $W = W_0 + \Delta W = W_0 + AB$
12 return Language model-initialized Prompt Decision Transformer;
```

5 testing tasks. The ant is also rewarded with high velocity along the goal direction. Different from segmenting the tasks by direction, Cheetah-vel penalizes the agent through the l_2 errors with the target velocities sampled from the velocity interval. There are 40 tasks with different goal velocities where 35 tasks are training tasks and 5 tasks are testing tasks. Except for the MuJoCo control meta-RL tasks, we also test our approach on Meta World [34] which is an open benchmark for meta-RL and multi-task learning. In this work, we evaluate our approach on Meta-World reach-v2 and Meta-World pick-place-v2. The objective of reach-v2 is to control the robot to reach the target position in 3D positions and pick-place-v2 is to grasp the object. Each task has a different goal position.

We utilize the dataset and settings from the Prompt-DT paper [31]. To be specific, the datasets of Cheetah-dir and Ant-dir come from the replay buffer of Soft Actor-Critic [7] and the dataset of Cheetah-vel comes from TD3 [5]. For Meta-World reach-v2 and Meta-World pick-place-v2, we collected the dataset through the expert policies provided in the open benchmark. We split the tasks in these environments into the training set and the testing set. The tasks in Cheetah-dir and Ant-dir are split by directions. The tasks in Cheetah-vel are split by the goal velocities. In Meta-World, the tasks are defined by different goal positions. The detailed task indexes can be found in Table 3. The experiments we conducted are all followed to this setting which guarantees consistency during the evaluation.

C.2 Baselines

Prompt-DT is the first method to utilize the prompt to guide Decision Transformer in testing with the few-shot demonstrations. Prompt-DT directly uses the prompt without any additional fine-tuning process when testing. Prompt-Tuning DT is based on Prompt-DT and utilizes prompt tuning methods when testing on the unseen task. Several prompt tuning techniques are used to tune the prompt to the specific target environment using preference ranking. Prompt Diffuser extends the prompt tuning method by leveraging diffusion models to generate high-quality prompts to improve the few-shot demonstration guidance. Beyond these baselines, Multi-Task Decision Transformer (MT-ORL) [3] is mentioned in Prompt-DT and Soft-Prompt [17] is described in Prompt Diffuser. So we do not demonstrate them in our experiments. For HDT [33], it utilizes the adapter to adapt the pre-trained model to new tasks, which is orthogonal to the prompt-based methods. Thus we do not include their results in our comparison.

Table 3: Training and testing task indexes when testing the generalization ability. We follow the tasks split between the Prompt-DT and previous works to guarantee a direct comparison with baselines.

Environment	Number of tasks	Tasks indexes
Cheetah-dir	Training set: 2 Testing set: 2	[0,1] [0,1]
Cheetah-vel	Training set: 35 Testing set: 5	[0-1,3-6,8-14,16-22,24-25,27-39{]} [2,7,15,23,26]
Ant-dir	Training set: 45 Testing set: 5	[0-5,7-16,18-22,24-29,31-40,42-49] [6,17,23,30,41]
Meta-World reach-v2	Training set: 15 Testing set: 5	[1-5,7,8,10-14,17-19] [6,9,15,16,20]
Meta-World pick-place-v2	Training set: 15 Testing set: 5	[0-10, 12-16,28-24,25-35,37-41,41-40] [11,17,25,36,41]

D Implementation

D.1 Implementation Details

In the empirical study, we implement our LPDT method with DistilGPT2 as the language initialization. The initialization language model weight comes from the Huggingface. The DistilGPT2 contains 82 million parameters which is lighter and more efficient than GPT2 with 124 million parameters. DistilGPT2 is pre-trained on the openwebtext [23] and is distilled by [26]. During the fine-tuning stage, we follow the same hyperparameters for Prompt-DT (see Appendix D.2 for detail). We also leverage the LoRA to highly reduce the parameters trained. For the prompt regularization, we use MLP to further encode the prompt embedding. For the supervised version of prompt regularization defined in (3.3), we directly use the logits from the MLP to compute the cross-entropy loss and refer to the method as LPDT-Classifier. For the unsupervised version of prompt regularization defined in (3.4), we calculate the similarity matrix through the cosine similarity based on the logits from the MLP and refer to it as LPDT-InfoNCE.

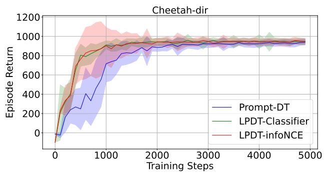
D.2 Hyperparameters

In this section, we show the hyperparameter of our LPDT conducted in Table 1. The hyperparameters have two parts which are the hyperparameters around the transformer and prompt regularization. We list these hyperparameters in Table 4.

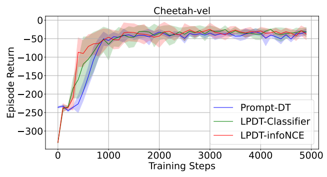
Table 4: Detail on hyperparameters used in our experiments in Table 1. We show that the hyperparameters in two parts which are parameters for model backbone and prompt regularization respectively.

Hyperparameters	Value
K (length of context τ)	20
Training batch size for each task	16
Number of evaluation episodes for each task	20
Learning rate	1e-4
Learning rate decay weight	1e-4
Language initialization	DistilGPT2
Embedding dimension	128
Activation	ReLU
Classifier hyperparameter	0.1
Classifier layers	2
Classifier MLP dimension	128
InfoNCE hyperparameter	0.1
InfoNCE temperature	1
InfoNCE MLP dimension	128

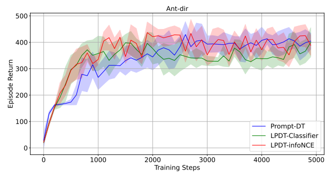
E Detailed Results



(a) Cheetah-dir



(b) Cheetah-vel



(c) Ant-dir

Figure 2: Results on MuJoCo controls with the Cheetah-dir, Cheetah-vel and Ant-dir for Prompt-DT and our two methods LPDT-Classifier and LPDT-InforNCE. The dataset we utilized is the full dataset. We plot the figures on unseen tasks with the average returns with one seed and 20 evaluation episodes. The figure demonstrates that our LPDT needs less sample data compared with Prompt-DT to achieve superior performance.

In this section, we provide comprehensive summaries of the experimental results. They are the results of all the unseen tasks in MuJoCo control environments. It includes all the experiments and ablation study results on our various components. We show that our methods with prompt regularization are much better than those without regularization and with text regularization. Table 5 demonstrates the results in Cheetah-dir, Table 6 refers to the results in Cheetah-vel and Table 7 refers to the Ant-dir. These results further support our observations and conclusions drawn in the experiment section.

Figure 2 shows that our approaches can outperform the baseline method Prompt-DT and demonstrate that they need fewer sample data compared with Prompt-DT to achieve superior performance. From Figure 2, we find that our approaches with language initialization can achieve competitive results with fewer training samples.

Table 5: Results on Cheetah-dir. The best mean scores are highlighted in bold. For each environment, prompt of the length of $K^* = 5$. The dataset we utilized is the full dataset. We test all the results on unseen tasks with three random seeds. Notably, our approach LPDT outperforms the baselines on the Cheetah-dir environment.

Methods	Cheetah-dir-0	Cheetah-dir-1	Average
Prompt-DT	669.79±5.68	1198.03±19.12	933.91±7.04
LPDT w/o regularization	686.19±3.30	1202.41±14.83	944.30±6.31
LPDT-Text	686.58±2.76	1201.61±1.57	944.09±1.68
LPDT-Classfer	692.63±3.57	1203.04±6.40	947.84±1.53
LPDT-InfoNCE	690.66±3.85	1212.78±5.26	**951.72±4.08**

Table 6: Results on the Cheetah-vel.The best mean scores are highlighted in bold. For each environment, prompt of the length of $K^* = 5$. The dataset we utilized is the full dataset.We test all the results on unseen tasks with three random seeds. Notably, our approach LPDT outperforms the baselines on the Cheetah-vel environment.

Methods	Cheetah-vel-2	Cheetah-vel-7	Cheetah-vel-15	Cheetah-vel-23	Cheetah-vel-26	Average
Prompt-DT	-54.48±1.76	-18.12±7.84	-35.92±6.22	-25.93±0.21	-39.11±5.31	-34.71±2.80
LPDT w/o regularization	-36.04±5.46	-18.06±5.40	-39.31±11.77	-51.87±1.55	-35.71±7.21	-36.20±4.05
LPDT-Text	-34.28±3.17	-18.01±11.37	-36.69±12.79	-53.91±3.45	-64.92±31.36	-41.56±9.65
LPDT-Classfer	-35.26±9.09	-13.86±5.02	-24.21±8.99	-48.05±13.96	-36.48±5.44	**-31.57±2.70**
LPDT-InfoNCE	-38.54±11.37	-13.11±4.30	-33.58±8.23	-37.79±2.19	-56.89±23.33	-35.98±7.15

F Ablation Studies

In this section, we provide ablation studies on LPDT to test the role of prompt regularization and language initialization respectively.

Table 7: Results on Ant-dir.The best mean scores are highlighted in bold. For each environment, prompt of the length of $K^* = 5$. The dataset we utilized is the full dataset. We test all the results on unseen tasks with three random seeds. Notably, our approach LPDT outperforms the baselines on the Ant-dir environment.

Methods	Ant-dir-6	Ant-dir-17	Ant-dir-23	Ant-dir-30	Ant-dir-41	Average
Prompt-DT	628.22±119.58	413.76±4.50	361.18±49.51	358.81±2.30	384.37±20.77	396.07±9.78
LPDT w/o regularization	362.08±64.18	411.79±6.13	324.34±98.67	381.20±1.79	325.36±1.84	360.95±11.07
LPDT-Text	254.53±18.46	418.98±8.00	362.91±54.60	371.94±4.79	348.19±46.48	351.31±23.13
LPDT-Classfer	398.80±120.62	417.85± 4.74	342.60± 82.08	365.02± 1.14	346.37±26.96	374.13± 23.04
LPDT-InfoNCE	555.18±164.12	418.42±5.43	376.34±15.52	362.91±5.22	349.52±46.83	**412.47**±**21.01**

Table 8: Results for MuJoCo control tasks and MW tasks with different regularization methods of our method including w/o regularization, text regularization, classifier regularization and InfoNCE regularization. The best mean scores are highlighted in bold. For each environment, the length of the prompt is $K^* = 5$. We test all the results on unseen tasks with three random seeds. The dataset we utilized is the full dataset. We demonstrate that the regularization on prompt can help distinguish the task and improve the performance compared with the method without regularization.

Task	Prompt-DT	LPDT w/o regularization	LPDT-Text	LPDT-classifier	LPDT-InfoNCE
Cheetah-dir	933.91±7.04	944.30±6.31	944.09±1.68	947.84±1.53	**951.72**±**4.08**
Cheetah-vel	-34.71±2.80	-36.20±4.05	-41.56±9.65	**-31.57**±**2.70**	-35.98±7.15
Ant-dir	396.07±9.78	360.95±11.07	351.31±23.13	374.13±23.05	**412.47**±**21.01**
MW reach-v2	**692.29**±**9.32**	431.87±115.19	459.65±76.01	497.61±48.15	528.21±114.24
MW pick-place-v2	**3773.82**± **356.05**	3700.34±68.45	3678.53±58	3508.12±243.93	3543.38±191.32

F.1 The role of prompt regularization

We first compare our proposed model with the same approach without the prompt regularization, denoted as LPDT w/o regularization. We also follow the settings of LaMo [27] and add a text regularization introduced by LaMo for comparison. We summarize the results in Table 8. Specifically, we find that prompt regularization helps the model distinguish between tasks and improve performance. LPDT without regularization or with text regularization suffers from inferior performance compared to our prompt regularization methods. Note that text regularization is trained with an NLP dataset, which is time-consuming and inefficient.

F.2 Data efficiency of language initialization

To verify whether language initialization can incorporate prior knowledge about the downstream RL tasks, we split the dataset to train Prompt-DT and our approaches. We also evaluate the results on a portion of the dataset using a ratio of 0.1. The results are summarized in Table 9.

Table 9: Ratio results for MuJoCo control tasks and MW tasks with the full dataset and 0.1 ratio (10%) dataset. The best mean scores are highlighted in bold. For each environment, the length of the prompt is $K^* = 5$. We test all the results on unseen tasks with three random seeds. We demonstrate that language initialization can improve the performance of our LPDT.

Task	Datset	Cheetah-dir	Cheetah-vel	Ant-dir	MW reach-v2	MW pick-place-v2
Prompt-DT	Full	933.91±7.04	-34.71±2.80	396.07±9.78	692.29±9.32	3773.82±356.05
	0.1	890.35±11.95	-42.46±9.25	361.13±4.46	**601.56**±**40.44**	3062.67±283.56
LPDT w/o regularization	Full	944.30±6.31	-36.20±4.05	360.95±11.07	431.87±115.19	3700.34±68.45
	0.1	927.80±4.91	-37.25±5.85	353.56±27.98	446.03±30.94	**3555.53**±**255.52**
LPDT-Classifier	Full	947.84±1.53	-31.57±2.70	374.13±23.05	497.61±48.15	3508.12±243.93
	0.1	**931.42**±**6.40**	-34.37±8.58	347.12±28.48	457.94±71.38	3118.57±77.47
LPDT-InfoNCE	Full	951.72±4.08	-35.98±7.15	412.47±21.01	528.21±114.24	3543.38±191.32
	0.1	919.75±4.17	**-34.20**±**8.21**	**369.73**±**21.97**	449.04±44.13	3341.35±131.93

Table 9 illustrates the results under different sizes of training datasets. We adopt DistilGPT2 for language initialization without any regularization. We also use LoRA to fine-tune the language models to the MuJoCo dataset. The results show that with only a 0.1 ratio (10%) of the dataset, the methods with language initialization outperform Prompt-DT when tested on unseen tasks, demonstrating that the language pre-trained model contains valuable knowledge for downstream RL tasks.