A Limitations

It is important to acknowledge the limitations of our work. On dense-reward *MuJoCo* tasks, we find that CQL is very competitive to LaMo, showing that value-based methods are still very strong in offline RL. Besides, the auxiliary language prediction loss in LaMo has only shown its advantage in very low-horzion tasks, *e.g.*, *Kitchen*, while in other tasks, it serves the purpose of preserving language capabilities but does not increase the performance significantly. How to better leverage the language reasoning ability to further help offline RL is thus a future direction. Lastly, limited by computational resources, we have not looked into utilizing larger language models (Touvron et al., 2023a;b; Chung et al., 2022), and we hope our work could motivate the community to explore further applications of LLMs in offline RL.

B Implementation Details

Codebase. Our codebase is mainly based on Chen et al. (2021) (`https://github.com/kzl/decision-transformer`) and Hu et al. (2023a) (`https://github.com/hukz18/DeFog`) with minimal modification on implementation details of original models. All codes of the baselines are directly from Tarasov et al. (2022) (`https://github.com/tinkoff-ai/CORL`) and Seno & Imai (2022) (`https://github.com/takuseno/d3rlpy`). Our official code is released at `https://github.com/srzer/LaMo-2023`.

Network architecture for LaMo. Except for the simplest task *Hopper*, where the observation space and action space of which is of only 11 and 3 dimensions respectively, we replace linear projections after the input with multi-layer perceptrons $M_{\hat{R}}, M_s, M_a$, and GELU (Hendrycks & Gimpel, 2016) as the activation function. With timesteps embedding $\omega(t)$, the embeddings of $\hat{R}_t, s_t, a_t$ are

$$u(x_t) = W_x^{(1)}\text{GELU}(W_x^{(0)}x_t) + \omega(t), x \in \{\hat{R}, s, a\}\,.$$

As for the Transformer, We mainly adopt the architecture of GPT-2 small model, with 124M parameters. The number of Transformer layers is 12, the number of attention heads is 12, and the hidden size is 768. Specifically, for *Kitchen*, when training on `Complete` (30%) dataset and `Partial` (100%) dataset, we empirically find that using GPT-2 medium[2], of which the number of layers is 24 and the hidden size is 1024, could enhance the performance.

C Dataset Descriptions

For *MuJoCo* and *Atari*, we mainly study the `Medium` dataset, generated by an agent trained using the SAC (Haarnoja et al., 2018) algorithm, which was terminated prematurely. The utilization of this dataset is aimed at minimizing variations in quality among different trajectories. The Atari datasets are taken from d4rl-atari (`https://github.com/takuseno/d4rl-atari`).

For *Kitchen*, we conduct experiments on both the `Complete` and the `Partial` dataset. In the `Complete` dataset, the robot performs all the required tasks sequentially, while the `Partial` dataset is composed of undirected data and ensures that a subset of the dataset can successfully solve the task.

For *Reacher2d*, which does not belong to *D4RL*, we train an agent of medium performance (average normalized score of 36.0 over 50 episodes) by PPO algorithm (Schulman et al., 2017), and then generate trajectories composed of 50 episodes simulated by that agent, referred to as `Medium` dataset.

To look into the low-data regime, we randomly downsample trajectories from the original dataset for a given sample ratio.

D Task Descriptions

Halfcheetah (MuJoCo): The goal is to make the cheetah move forward (right) as fast as possible by applying torque to its joints.

[2]`https://huggingface.co/gpt2-medium`

Hopper (MuJoCo): The goal is to achieve forward (right) motion through controlled hops.

Walker2d (MuJoCo): The goal is to coordinate movements to achieve forward (right) direction.

Reacher2d: The goal is to move the robot's end effector (fingertip) close to a randomly spawned target.

For *Reacher2d*, We compute the average performance over the last 12.5K training steps out of a total of 37.5K training steps with evaluations conducted every 2500 training steps.

Kitchen: The objective in each task is to interact with items to reach a specific desired configuration.

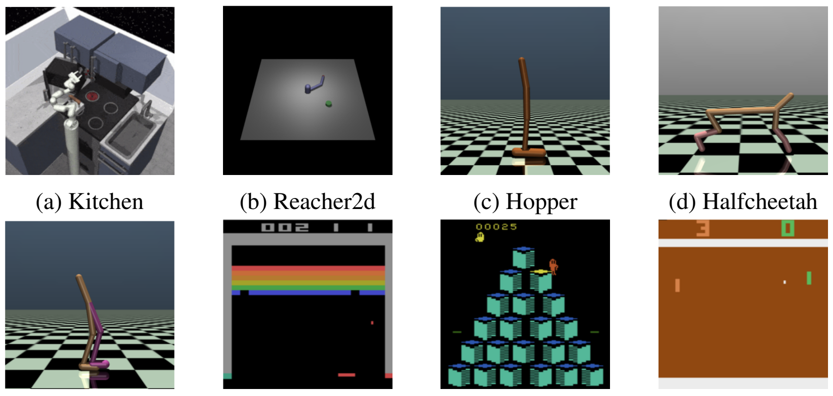
Breakout (Atari): Players control a paddle to hit a ball at a brick wall, aiming to break it down. Players have five lives.

Qbert (Atari): The objective is to change the color of cubes on the pyramid to match the 'destination' color by hopping on each cube while avoiding obstacles.

For *Breakout*, on which algorithms converge fast, we compute the average performance over the last 10K training steps out of a total of 50K training steps with evaluations conducted every 2500 training steps.

Pong (Atari): Players compete to deflect the ball away from their goal and into the opponent's goal using paddles.

In Figure 8, we provide visualizations of each task.



(a) Kitchen (b) Reacher2d (c) Hopper (d) Halfcheetah

(e) Walker2d (f) Breakout (g) Qbert (h) Pong

Figure 8: **Visualization of Tasks** from 3 domains: *Kitchen*, *MuJoCo*, and *Atari*.

E TASK SCORE NORMALIZATION

Task Name	Random Score	Expert Score
Kitchen	0	4
Reacher2d	0	100
Hopper	−20.3	3234.3
HalfCheetah	−280.2	12 135.0
Walker2d	1.6	4592.3
Breakout	1.7	31.8
Qbert	163.9	13 455.0
Pong	−20.7	9.3

Table 5: **Scores used for normalization.** Scores of each task are linearly normalized by the corresponding random score and expert score.

The scores we present are normalized using the formula:

$$\text{normalized score} = \frac{\text{score} - \text{random score}}{\text{expert score} - \text{random score}} \times 100\,,$$

where the random scores and the expert scores are provided in Table 5, so that 100 represents the expert score and and 0 represents the score of a random policy, following the protocols of Fu et al. (2020) and Hafner et al. (2020).

F HYPERPARAMETERS

In Table 6 and Table 7, we list the task-specific hyperparameters and task-agnostic hyperparameters, respectively. More details can be referred to `https://github.com/srzer/LaMo-2023`.

Task Name / Variable	Learning Rate	Weight Decay	Context Length	Return-to-go	Training Steps
Kitchen	1×10^{-4}	1×10^{-5}	20	$3, 4, 5$	100K
Reacher2d	1×10^{-5}	1×10^{-4}	5	$40, 76$	100K
Hopper	1×10^{-4}	1×10^{-5}	20	$1800, 2200, 3600$	100K
HalfCheetah	1×10^{-4}	1×10^{-5}	20	$6000, 8000, 12000$	100K
Walker2d	1×10^{-5}	1×10^{-4}	20	$2500, 4000, 5000$	100K
Breakout	1×10^{-3}	1×10^{-2}	30	$90, 120$	50K
Qbert	1×10^{-3}	1×10^{-5}	30	14000	100K
Pong	3×10^{-4}	1×10^{-1}	30	$10, 20$	100K

Table 6: **Task-Specific Hyperparameters.**

Variable	Value
Number of Transformer Layers	12
Number of MLP Layers (Kitchen & MuJoCo)	3
Number of CNN Layers (Atari)	3
Number of CNN Channels (Atari)	$32, 64, 64$
Dimension of CNN Kernels (Atari)	$8, 4, 3$
Hidden Dimension	768
LoRA Rank	16 (Kitchen & MuJoCo), 32 (Atari)
Batch Size	64 (Kitchen & MuJoCo), 128 (Atari)
Dropout	0.1

Table 7: **Task-Agnostic Hyperparameters.**

We follow the common practice of DT that uses multiple values as rtg, and reports the best one, as shown in `https://github.com/kzl/decision-transformer`. The values we adopt are validated by experiments. And one of the reasons is that for different sampling ratio, the best rtg would be different.

G MORE RESULTS

Language ability test of models. With the prefix prompt *Hello, my name is Tom.*, answers of different models are:

- GPT-2: *I'm not very funny anymore, and I'm not a comedian, and neither am I a guy. Oh, I'd like to do some comedy so that we could work together and actually work out some pretty big*
- Early-ended Pre-trained: *Andriaki. = = Reception = = = Critical response = = = A number of reviewers praised Tom McNeill's performance, commenting that*
- Random Corpus: *The, was of the from on to in @ the on the to for, the, and to that =.. by of for. on that the ' that the*

Results on datasets with varying qualities. Tables 8 and 9 present testing results when the models are trained on datasets of different qualities, `Medium-Expert` and `Medium-Replay`. LaMo shows competitive performance over the baselines, especially on `Medium-Replay` (1%) datasets.

Results on more tasks. In Tables 10 and 11, we present additional results for tasks in *Atari* and *MuJoCo*. Specifically, in the *Freeway* and *Asterix* tasks, LaMo demonstrates significant advancements over DT and Wiki-RL, effectively narrowing the performance disparity with value-based

Task	Dataset	Ratio	Ours	DT	CQL	IQL	TD3+BC
Hopper	Med-Replay	0.01	29.4 $\pm$ 6.3	29.3 $\pm$ 4.0	1.3 $\pm$ 1.3	16.3 $\pm$ 2.3	19.7 $\pm$ 2.1
Halfcheetah	Med-Replay	0.01	14.6 $\pm$ 4.5	10.0 $\pm$ 2.6	-0.2 $\pm$ 0.2	17.8 $\pm$ 4.5	7.4 $\pm$ 2.7
Walker2d	Med-Replay	0.01	28.1 $\pm$ 1.0	12.4 $\pm$ 1.3	31.3 $\pm$ 2.6	30.2 $\pm$ 1.2	11.0 $\pm$ 1.3
Average			24.0(↑40%)	17.2	10.8	21.4	12.7

Table 8: **Normalized score on Medium-Replay Dataset**. Blue highlight indicates the highest score, orange highlight indicates the second-highest score.

Task	Dataset	Ratio	Ours	DT	Wiki-RL	CQL	IQL	TD3+BC	BC
Hopper	Med-Expert	1	109.9 $\pm$ 1.4	107.6 $\pm$ –	110.9 $\pm$ –	105.4 $\pm$ –	91.5 $\pm$ –	98.0 $\pm$ –	52.5 $\pm$ –
Halfcheetah	Med-Expert	1	92.2 $\pm$ 0.7	86.8 $\pm$ –	91.8 $\pm$ –	91.6 $\pm$ –	86.7 $\pm$ –	90.7 $\pm$ –	55.2 $\pm$ –
Walker2d	Med-Expert	1	108.3 $\pm$ 1.1	108.1 $\pm$ –	108.9 $\pm$ –	108.8 $\pm$ –	109.6 $\pm$ –	110.1 $\pm$ –	107.5 $\pm$ –
Average			103.5(↑3%)	100.8	103.9	101.9	95.9	99.6	71.7

Table 9: **Nomalized score on Medium-Expert Dataset**. Values without standard deviations are taken from Kostrikov et al. (2022).

based methodologies. Furthermore, in the *Ant* task, LaMo surpasses the baseline scores, indicating a notable improvement in performance.

Task	Dataset	Ratio	Ours	DT	Wiki-RL	CQL	BCQ	NFQ
Freeway	Medium	0.1	83.0 $\pm$ 1.7	70.1 $\pm$ 4.0	72.0 $\pm$ 3.9	101.2 $\pm$ 1.7	79.1 $\pm$ 6.1	56.2 $\pm$ 19.1
Asterix	Medium	0.1	2.8 $\pm$ 1.0	0.7 $\pm$ 0.7	0.3 $\pm$ 0.4	4.3 $\pm$ 1.0	1.9 $\pm$ 0.5	-0.1 $\pm$ 0.0
Average			42.9(↑21%)	35.4	36.2	52.7	40.5	28.1

Table 10: **More tasks in Atari**. We conduct experiments and report normalize scores on another 2 games, *Freeway* and *Asterix*, in the domain of *Atari*, to present the remarkable improvement of LaMo over DT and Wiki-RL.

Comparison with Diffusion-QL. In Table 12, a comparative analysis is presented between our method (LaMo) and the recent powerful diffusion policy, Diffusion Q-learning (Diffusion-QL) (Wang et al., 2023c), across three distinct tasks. Notably, LaMo outperforms Diffusion-QL in all three tasks, with particularly remarkable results in the low-data regime.

Overfitting issues of CQL. In *Kitchen* tasks, CQL faces the issue of overfitting, causing a notable drop in its performance, as presented in Figure 9.

Results based on top-k metric. We provide the experiment results of *Kitchen* using top-k metric, which calculates the average scores over the k checkpoints with the highest testing scores. As is shown in Table 13, LaMo still outperforms other DT-based methods and value-based methods.

Effects of the model size. In Figure 10, the influence of language model size on the performance of LaMo is depicted. The results indicate that GPT2-small already achieves satisfactory performance. Additionally, in specific tasks, GPT2-medium demonstrates a slight edge over GPT2-small, showcasing incremental improvements with increased model size.

Hyperparameter tuning for baselines. As demonstrated in Figure 11, we conduct hyperparameter tuning for both DT-based and value-based baselines. As for Wiki-RL, we sincerely follow the authors' statements in their paper (Reid et al., 2022) Section 4.1, to use the same learning hyperparameters as DT. These results verify the baselines' performance reported in our study.

Number of parameters of our method and baselines. We have presented Table 14 listing the trainable parameter sizes of LaMo alongside various baselines. For value-based methods, section 3 and the ablation study in section 4.5 in Tarasov et al. (2023) demonstrate that deepening the network structure does not yield improvements, and our experiments shown in Figure 12 reveal a similar trend for increasing the network's width. Thus we adhere to widely accepted model sizes, which are much smaller than Transformer-based methods. Besides, it is important to emphasize that simply increasing the size of the Transformer will not boost the performance, as shown in the results of Reid et al. (2022) and our ablation studies. Moreover, although our method involves a relatively large model, the number of trainable parameters is fairly small, which is 3.5M, comparable with 7.3M

Task	Dataset	Ratio	Ours	DT	Wiki-RL	CQL	IQL	TD3+BC
Ant	Medium	0.1	87.8 ± 4.5	87.2 ± 4.6	70.5 ± 4.9	95.6 ± 4.9	92.2 ± 6.4	97.5 ± 5.4
Ant	Medium	0.01	86.3 ± 6.2	77.8 ± 4.8	65.9 ± 6.1	76.2 ± 10.5	46.4 ± 5.5	3.4 ± 1.8
Ant	Medium	0.005	90.2 ± 3.7	76.5 ± 4.9	71.3 ± 5.8	64.5 ± 6.3	65.8 ± 6.3	1.2 ± 3.4
	Average		88.1(↑9%)	80.5	69.2	78.8	68.1	34.0

Table 11: **More tasks in MuJoCo**. We conduct experiments and report normalized scores on another environment, *Ant*, in the domain of *MuJoCo*, to present the competitive performance of LaMo.

Task	Dataset	Ratio	LaMo	Diffusion-QL
Hopper	Medium	0.005	57.0	13.2
Hopper	Medium	0.01	52.0	35.7
Hopper	Medium	0.1	73.7	68.4
Halfcheetah	Medium	0.005	39.0	36.5
Halfcheetah	Medium	0.01	40.6	34.8
Halfcheetah	Medium	0.1	42.1	46.8
Walker2d	Medium	0.005	66.9	32.3
Walker2d	Medium	0.01	74.5	44.7
Walker2d	Medium	0.1	70.4	55.2
	Average		57.4	40.8

Table 12: **Comparing LaMo with Diffusion-QL**. We compare LaMo with the recent strong baseline, Diffusion Q-learning (Diffusion-QL) (Wang et al., 2023c)) across 3 different tasks.

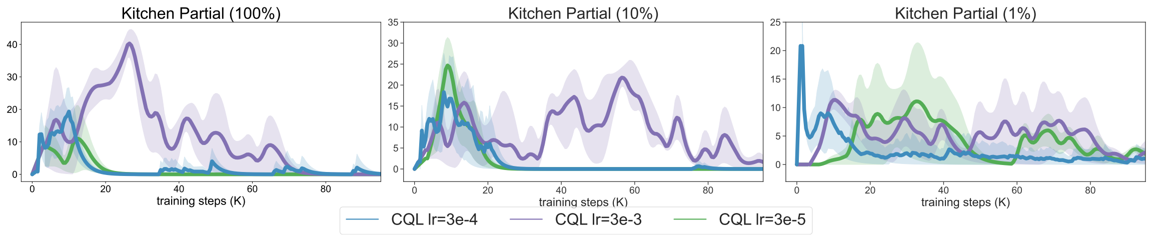


Figure 9: **Normalized score of CQL**. CQL consistently suffers from severe overfitting in *Kitchen* tasks with various learning rates.

Task	Dataset	Ratio	LaMo	DT	Wiki-RL	CQL	IQL	TD3+BC	BC
Kitchen	Partial	1	56.4	59.8	39.9	42.5	56.3	17.1	33.8
Kitchen	Complete	1	73.3	60.6	35	45.7	49.4	40.3	33.8
	Average		64.9(↑8%)	60.2	37.5	44.1	52.8	28.7	33.8

Table 13: **Normalized score of Kitchen based on the metric of top-3 performance**. We present the average normalized scores over the best 3 checkpoints in each run. The results agree with those in D4RL (Fu et al., 2020). Blue highlight indicates the highest scores, and red numbers represent the improvement of LaMo over DT.

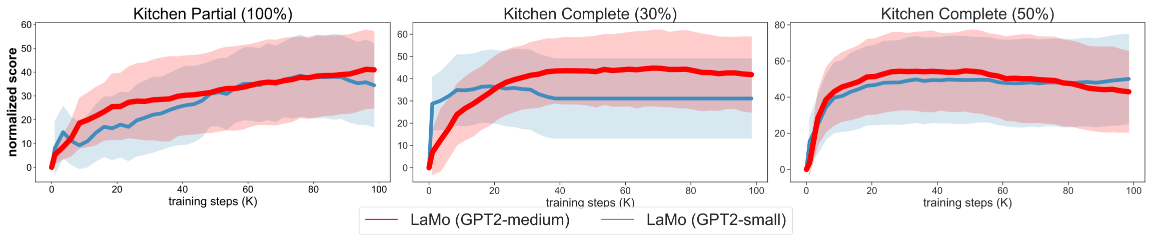


Figure 10: **Effects of GPT-2 model size**. We adopt two language model backbones of different size, GPT2-small[1] and GPT2-medium[2]. GPT2-small already shows satisfying performance. In Kitchen Complete (30%) and Kitchen Partial (100%), GPT2-medium slightly exceeds GPT2-small.

[1] `https://huggingface.co/gpt2` [2] `https://huggingface.co/gpt2-medium`

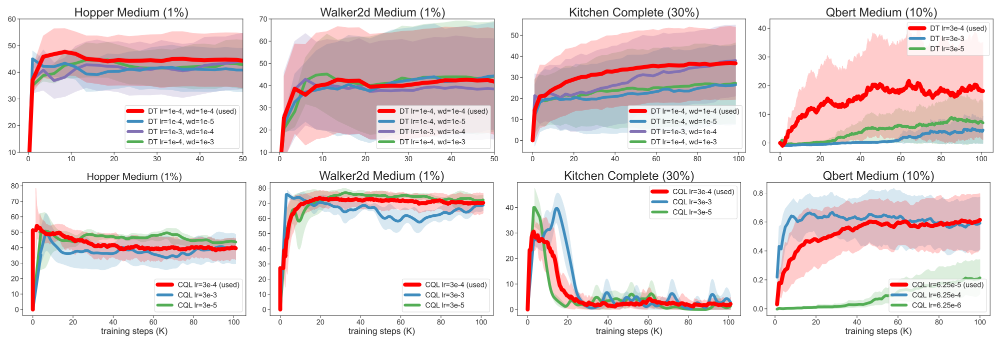


Figure 11: **Baseline hyperparemeter tuning.** We tune the hyperparameters of two strong baseline methods, DT and CQL. We compare the experiment results with different learning rate (lr) and weight decay (wd) across 3 domains, *MuJoCo*, *Kitchen* and *Atari*.

of Decision Transformer. So the difference of the number of parameters between Transformers and value-based methods does not compromise the fairness of the comparisons of performance.

Algorithms	Total Parameters	Trainable parameters
LaMo	128M	3.5M
Wiki-RL	125M	125M
DT	7.3M	7.3M
CQL	0.41M	0.41M
IQL	0.33M	0.33M
TD3+BC	0.25M	0.25M

Table 14: **Number of parameters of each algorithms**. We conduct a comparative analysis of the total parameter sizes and trainable parameter sizes of LaMo and various baseline.

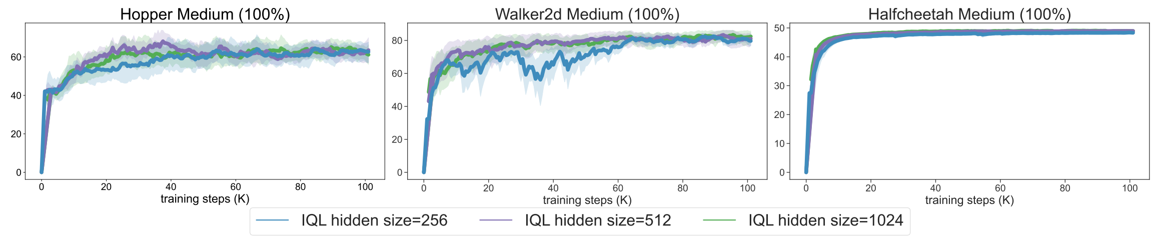


Figure 12: **Effects of the width of networks on the performance of value-based methods**. We train IQL with different hidden size across 3 different tasks, demonstrating that increasing the width does not yield improvements.

H The Surprising Effectiveness of Frozen Random Weights with LoRA

To show the help of the pre-trained weight of the LLM, we actively run ablation experiments. The only difference is that the weights of the Transformer layers are randomly initialized according to the intialization of GPT2 (Radford et al., 2019). We present the results in Table 15 and Table 16. We observe that LoRA harnesses the potential of the pre-trained model, enabling it to outperform DT significantly in both sparse-reward and dense-reward tasks. Besides, the pre-trained model outperforms the randomly initialized model noticeably.

Although we do not utilize a feature-aligned functor for the two domains, language and motion control, in LaMo's pipeline, which could be a promising future work, we hypothesize that freezing most parameters and training on language tasks simultaneously could potentially force the feature alignment between two domains, and thus the general knowledge of sequential modelling could be transferred. Besides, Aghajanyan et al. (2021)'s theoretical analysis also backs our claims, which shows the connection between intrinsic dimension and generalization.

Task	Dataset	Ratio	LaMo	LaMo w/o. PT	DT
Kitchen	Partial	0.01	11.6 $\pm$ 3.0	9.8 $\pm$ 2.6	0.9 $\pm$ 0.9
Kitchen	Partial	0.1	35.1 $\pm$ 5.2	24.8 $\pm$ 4.3	22.6 $\pm$ 6.8
Kitchen	Partial	1	46.6 $\pm$ 5.3	43.6 $\pm$ 7.4	33.8 $\pm$ 14.5
Kitchen	Complete	0.3	45.9 $\pm$ 2.9	42.4 $\pm$ 4.7	31.5 $\pm$ 4.5
Kitchen	Complete	0.5	50.6 $\pm$ 6.1	56.8 $\pm$ 4.5	36.6 $\pm$ 5.1
Kitchen	Complete	1	64.2 $\pm$ 5.3	51.8 $\pm$ 3.7	52.8 $\pm$ 3.7
Reacher2d	Medium	0.1	12.4 $\pm$ 3.8	6.8 $\pm$ 3.6	2.3 $\pm$ 1.5
Reacher2d	Medium	0.3	31.2 $\pm$ 7.6	20.2 $\pm$ 5.4	6.4 $\pm$ 2.6
Reacher2d	Medium	1	33.0 $\pm$ 8.3	28.2 $\pm$ 8.4	22.8 $\pm$ 6.0
Average			36.7	31.6	23.3

Table 15: **Ablation on the effectiveness of pre-training for** 2 **sparse-reward tasks**. We replace the pre-trained LM in LaMo with a randomly initialized model of the same structure, denoted as *LaMo w/o. PT*. We compare LaMo with *LaMo w/o. PT* and DT. Blue highlight indicates the highest score.

Task	Dataset	Ratio	LaMo	LaMo w/o. PT	DT
Hopper	Medium	0.005	57.0 $\pm$ 7.1	43.5 $\pm$ 2.8	35.8 $\pm$ 6.6
Hopper	Medium	0.01	52.0 $\pm$ 4.6	46.0 $\pm$ 3.1	41.9 $\pm$ 5.2
Hopper	Medium	0.1	73.7 $\pm$ 3.5	57.7 $\pm$ 2.6	57.3 $\pm$ 3.8
Hopper	Medium	1	74.1 $\pm$ 5.3	64.5 $\pm$ 4.9	60.9 $\pm$ 3.3
Halfcheetah	Medium	0.005	39.0 $\pm$ 1.6	39.2 $\pm$ 1.3	22.4 $\pm$ 5.2
Halfcheetah	Medium	0.01	40.6 $\pm$ 1.3	40.6 $\pm$ 1.4	29.6 $\pm$ 4.8
Halfcheetah	Medium	0.1	42.1 $\pm$ 0.6	41.4 $\pm$ 0.7	41.7 $\pm$ 0.8
Halfcheetah	Medium	1	42.5 $\pm$ 0.4	42.8 $\pm$ 0.3	42.6 $\pm$ 0.5
Walker2d	Medium	0.005	66.9 $\pm$ 5.4	57.0 $\pm$ 5.8	16.7 $\pm$ 4.8
Walker2d	Medium	0.01	74.5 $\pm$ 4.7	74.2 $\pm$ 1.9	38.9 $\pm$ 9.3
Walker2d	Medium	0.1	70.4 $\pm$ 4.2	70.9 $\pm$ 4.0	70.2 $\pm$ 7.5
Walker2d	Medium	1	73.3 $\pm$ 3.1	69.8 $\pm$ 9.3	70.2 $\pm$ 4.3
Average			58.8	54.0	44.0

Table 16: **Ablation on the effectiveness of pre-training for** 3 **dense-reward tasks**. Blue highlight indicates the significantly highest score.

Another interesting phenomenon is that even without pre-training, the model, with enlarged model size, deepened embeddings and LoRA adapting techniques, could still reach higher performance than original DT. Shen et al. (2021) observes the same results, while they use randomly initialized frozen layers as a cheap way to deepen their model and achieve better performance, albeit their application is in the field of natural language. Jarrett et al. (2009) also proposes Random Filter for object detection, and it has decent performance as a feature extractor.

Although we have not looked into the effectiveness of random initialized Transformers deeply, based on the experiment results and previous works, we hold a belief that freezing random weights with minimal adaption is a promising approach.

I Exploratory Findings and Additional Discussion

I.1 Larger Language Models

In this work, we treat the size of GPT-2 as hyperparameters and observed that in most cases, they did not exhibit significant differences in performance. Furthermore, we explore the pre-trained model LLaMA and apply various techniques, including LLaMA adaptor (Zhang et al., 2023b), lora, and joint training. However, we find that none of these approaches demonstrates substantial improvements over the baseline. Constrained by computational resources, we are unable to delve further into this area, but it could serve as an intriguing avenue for future research endeavors.

I.2 Other Embedding Methods

We have tried another embedding method, which is inspired by RT-2 (Brohan et al., 2023a). We are intended to stimulate the power of pre-trained LM in a way that is closer to processing language. We discretize each dimension of each vector into 512 bins after normalization and thus get the index sequence. Indexes are understood by the model as tokens, and the corresponding embeddings of indexes are set independently for returns-to-go, states and actions, while all of them are intialized as pre-trained embeddings for those digit tokens.

While this approach works well in situations with 100% data availability, it exhibits poor performance in low-data regime, even when various techniques are employed. We suspect that the limitation may stem from the lack of generalizability in this method. In the future, addressing this issue could involve exploring higher-level representation learning techniques to improve performance.