Masked Autoencoding for Scalable and Generalizable Decision Making

Fangchen Liu * UC Berkeley
Hao Liu * UC Berkeley
Aditya Grover UCLA
Pieter Abbeel UC Berkeley

Abstract

We are interested in learning scalable agents for reinforcement learning that can learn from large-scale, diverse sequential data similar to current large vision and language models. To this end, this paper presents masked decision prediction (MaskDP), a simple and scalable self-supervised pretraining method for reinforcement learning (RL) and behavioral cloning (BC). In our MaskDP approach, we employ a masked autoencoder (MAE) to state-action trajectories, wherein we randomly mask state and action tokens and reconstruct the missing data. By doing so, the model is required to infer masked out states and actions and extract information about dynamics. We find that masking different proportions of the input sequence significantly helps with learning a better model that generalizes well to multiple downstream tasks. In our empirical study we find that a MaskDP model gains the capability of zero-shot transfer to new BC tasks, such as single and multiple goal reaching, and it can zero-shot infer skills from a few example transitions. In addition, MaskDP transfers well to offline RL and shows promising scaling behavior w.r.t. to model size. It is amenable to data efficient finetuning, achieving competitive results with prior methods based on autoregressive pretraining. We have open-sourced the implementation of our method at `https://github.com/FangchenLiu/MaskDP_public`.

1 Introduction

Self-supervised pretraining has made tremendous successes for unsupervised representation learning in natural language processing (NLP) and vision [12, 9, 3, 4]. These methods work by predicting a removed portion of the data, which is often referred to as masked token prediction. By varying the masking patterns and architectures, different methods have been developed for NLP and vision, $e.g.$, Transformer [33], GPT [4], BERT [9], and MAE [12]. These methods are simple to implement and scalable to large Internet-scale datasets and deep neural networks, leading to excellent flexibility and generalization for downstream tasks [9, 12, 1].

In this work, we explore the generality of masked token prediction for generalizable and flexible reinforcement learning (RL). Prior work has explored sequence modeling for sequential decision making in the context of offline RL $e.g.$ decision transformer (DT) [6] and trajectory transformer (TT) [13], and black-box optimization $e.g.$ transformer neural processes (TNP) [19]. These methods are based on autoregressive next token prediction, similar to GPT [4]. While promising, these works do not leverage diverse unlabeled data for generalization across various downstream tasks. In addition, DT [6] needs reward-labeled high quality datasets, while TT [13] requires discretizing states and actions, further limiting its applicability. The flexibility of applying arbitrary masks for executing various task specifications in RL significantly lags behind NLP and vision.

*equal contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

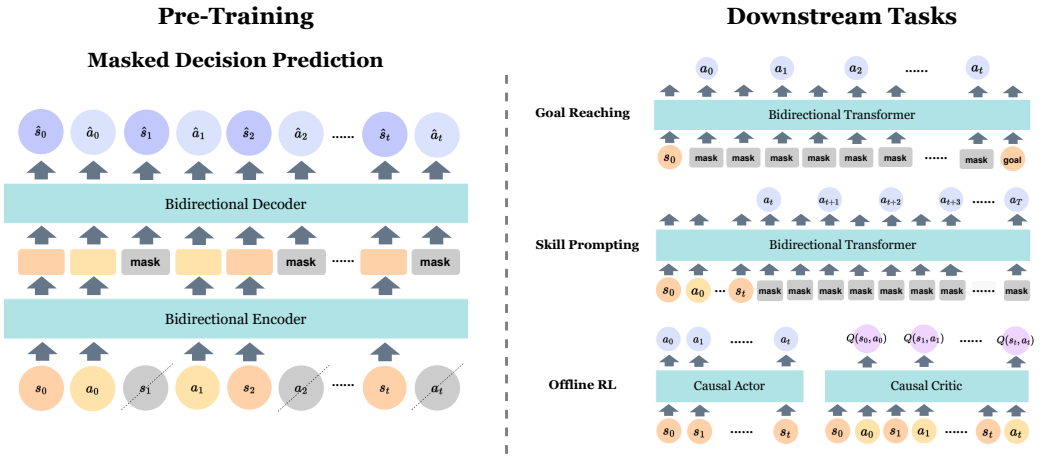

Figure 1: Illustration of MaskDP. During pretraining stage, we perform the masked token prediction task. And after pretraining, the model can be deployed to various downstream tasks using different mask patterns.

We propose Masked Decision Prediction (MaskDP), a pretraining method to learn generalizable models that achieve data-efficient adaptation to various downstream tasks. MaskDP is a self-supervised pretraining method that can leverage unlabeled diverse data. With MaskDP pretraining, the model can generalize well to both goal reaching and offline RL, two distinctive and popular RL paradigms.

Our first key observation is that masked token prediction with random masking similar to BERT [9] and MAE [12] provides a general and flexible way for learning from unsupervised data. Unlike autoregressive action prediction used in prior works, random masking is strictly more general and requires the model to infer masked out states and actions, and thus leads to a single model that can reason about both the forward and inverse dynamics from each sample.

Our second key observation is that since states and actions are highly correlated temporally, trajectories have significantly lower information density, $i.e$, it is easier to predict action or state based on nearby states and actions. Consequently, a high mask ratio (95%) is necessary to make reconstruction task meaningful. Unlike in MAE [12] and BERT [9] where the goal is learning representations, we want to directly apply MaskDP to various downstream tasks, and different mask ratios induce different pre-train and downstream gaps. For example, consider the goal-reaching task within certain time limit. Given current state, future goal and mask tokens between them, the model should be able to inpaint intermediate actions as the goal-reaching plan. The mask ratio varies from short-term plans to long-term plans. Therefore, we combine multiple different mask ratios ($e.g.$ 15%, 35%, 75%, and 95%), and mask a portion of data using a randomly sampled mask ratio. Our experiments show that doing so is crucial to achieving high performance. We show that self-supervised pretrained MaskDP achieves high performance in challenging multiple goals reaching setting, outperforming strong baselines in a zero-shot manner.

We highlight our key results here:

- **Single goal reaching**: MaskDP achieves performance that exceeds or matches both training from scratch task-specific methods and other pretraining based methods.
- **Sequential multiple goal reaching**: MaskDP can reach a sequence of goals effectively, even without closed-loop execution, while outperforming iterative baselines significantly.
- **Offline RL**: MaskDP achieves competitive results as specialized approaches. Notably, we demonstrate that non-autoregressive architecture works well for offline RL tasks.

2 Related work

Masked modeling in language and vision. Large-scale language models are highly successful [9, 4]–after pretraining on a large amount of data, these pretrained representations generalize well to various downstream tasks. Taking inspiration from the success in NLP, Transformer [33] based

methods have been proposed to model images [7, 10, 3, 12]. iGPT [7] operates on sequences of pixels and predicts unknown pixels. BEiT [3] proposes to predict discrete tokens [32, 22]. MAE [12] proposes to randomly mask patches of the input image and reconstruct the missing pixels. Since we apply random mask across states and actions, our work is also related to prior work on masked prediction across multiple input modalities [see e.g. 34].

Masked modeling in RL Masking trajectories for pretraining has been studied in RL [21, 23, 6, 13, 39]. MVP [35] studies transferring pretrained visual representations to RL tasks. MWM [25] studies masked prediction over convolutional features, and learn a latent dynamics model. Modelling inverse dynamics has also been studied for robot learning from demonstrations and sim-to-real transfer [8, 30]. TT [13] studies autoregressive next token prediction for model-based RL applications. DT [6, 39] study masking autoregressive next token prediction conditioned on return. TNP [19] and BONET [14] study autoregressive masking for sequential decision making for black-box optimization. ICM [21] and SPR [23] study predicting masked state and action in a transition tuple for exploration. Different from these works, MaskDP randomly masks a portion of trajectories and generalizes prior masking strategies such as inverse dynamics. In addition, MaskDP generalizes well to downstream tasks while prior work is task-specific. Concurrent to our work, UniMASK [5] proposes using bidirectional transformer to predict randomly masked state and action tokens and demonstrates that the resulting model performs well on a large variety of tasks. UniMASK shows leveraging a single set of weights for various tasks using a task-agnostic architecture, while our work focuses achieving state-of-the-art performance on a range of downstream tasks with minimal task specific designs.

Unsupervised pretraining in RL Our work falls under the category of self-supervised pretraining in RL. Self-supervised discovery of a set of task-agnostic behaviors by means of seeking to maximize an intrinsic reward has been explored as intrinsic motivation [2], often with the goal of encouraging exploration [27, 20]. APT [18] studies nonparametric entropy maximization for pretraining and is extended to learning skills [17]. Proto-RL [36] further improves pretraining by representation learning. CIC [16] combines contrastive learning with skill discover and improves results on URL benchmarks [15]. APV [26] shows successful transfer of pretrained representation across domains. Many of these methods are used to pretrain agents that are later adapted to specific reinforcement learning tasks. Using offline data for pretraining agents has also been explored in prior work [24, 28, 38]. SGI [24] proposes combining self-predictive representation [23] and inverse dynamics prediction. ATC [28] studies contrastive pretraining on trajectories and shows transferring the representations to downstream tasks.

3 Method

The key idea in MaskDP is to mask and reconstruct state-action sequences during the pretraining stage. Post pretraining, MaskDP can be zero-shot deployed or finetuned for various downstream tasks. The paradigm of the model for pretraining and finetuning is summarized in Figure 1.

3.1 MaskDP Pretraining

Random masking. For sequences with low information density, a *high* masking ratio is typically applied to eliminate information redundancy and make the task sufficiently difficult to avoid trivial interpolation from visible neighbor tokens. However, unlike vision and language, where the goal is to learn good representations; we also consider directly deploying this model by leveraging its inpainting capability for various downstream tasks. For example, we can give the model a goal at timestep T and mask all the future inputs, the model can generate intermediate actions by inpainting the mask tokens. The mask ratio varies from goal to goal, depending on the time budget. To reduce the gap between training and deployment, we keep a set of mask ratios ($i.e.$ 15%, 35%, 50%, 75%, and 95%), and the data is randomly masked with a ratio sampled from this set. We find that masking multiple proportions of the input yields a meaningful self-supervisory task.

We apply random masking on state tokens and action tokens independently. By doing so, the model is implicitly learning both forward and inverse dynamics. This also provides more flexibility as we can provide state or action-level inputs but not transition-level.

Architecture Our encoder is a Transformer [33] but applied only on visible, unmasked states and actions, similar to MAE [12]. The states and actions are first embedded by separated linear layers, positional embeddings are then added, and lastly, the embeddings are processed by a series of self-attentional blocks. The decoder operates on the full set of encoded visible state and action tokens and mask tokens. Each mask token is a shared, learned vector that indicates the presence of a missing token to be predicted. Similar to the encoder, the masked whole sequence will pass through separated linear projections added with positional embedding prior to being passed to the decoder. Both the encoder and decoder are bidirectional.

Prediction target Our MaskDP reconstructs the input by predicting the whole action and state sequences. The last layer of the decoder consists of two MLPs to decode states and actions separately. The loss function computes the mean squared error (MSE) between the reconstructed whole sequence and original inputs. Different from other masked prediction variants [12, 9], we found mask loss is not useful in our setting, as our goal is to obtain an scalable decision making model but not only for representation learning.

3.2 MaskDP Downstream Tasks

MaskDP for goal reaching We consider the problem of reaching one goal or multiple goals from a given state. The model has to generate a sequence of actions to reach goals within a certain amount of steps. MaskDP denoising pretraining objective fits the goal reaching scenario well as the model must learn to inpaint masked actions based on remaining states. In this task, the MaskDP encoder input is a concatenation of initial state and goals, and the decoder input is a concatenation of initial state embedding, masked token sequence, and goal embeddings. Note that the number of masked tokens determines the number of timesteps the model is expected to reach the given goals. The model then generates a state-action sequence, where we can directly execute the whole action sequence (namely "open-loop"), or only execute the first action and forward the model again with the obtained new observation (namely "closed-loop").

MaskDP for skill prompting Skill prompting requires the model to generate a trajectory conditioned a given context. For example, consider a walker agent: if we prompt it with a few state-action pairs of walking/running/standing, it should continue to generate a trajectory in the same skill pattern. Accordingly, we append the observed initial state-action sequences with masked tokens for the future. The model can be rolled out once to generate the whole future sequence, or queried repeatedly to refill the masked tokens at each time step. Similar to goal-reaching task, we refer to these strategies as "open-loop" and "closed-loop" respectively.

MaskDP for offline RL In offline RL, the objective is to learn one model for maximizing the return for a task specified by a reward function. This is different from our self-supervised pretraining target, so extra finetuning is needed. We adopt a standard actor-critic framework similar to TD3 [11] by adding a critic head and actor head, where the actor takes a state sequence as input, and the critic takes the state-action sequence as input. Both are mask-free. To match the setting in RL, we change the bidirectional attention mask in the transformer to a causal attention mask. More details about RL finetuning can be found in section 4.2.3.

4 Experiments

In our experiments, we evaluate transfer learning in downstream tasks using MaskDP. Section 4.1 introduces the environments, pretraining, and the baselines compared in experiments. Section 4.2 summarizes the results of MaskDP on goal reaching, skill prompting, and offline RL. Through further analysis in Section 4.3, we present an ablation study on various design choices of our model.

4.1 Experiments Setup

Environments: domains vs. tasks We adopt the environment setup used in EXoRL [37], based on DeepMind control suite [29], where a domain describes the type of agent (e.g. Walker) but tasks are specified by rewards (e.g., Walker walk, Walker run). We use 3 domains (Walker, Cheetah and Quadruped) with 7 tasks in total. More details about the environments can be found in the Appendix.

Pretraining datasets Real-world pretraining data generally varies greatly in quality. To mimic this, we construct two different pretraining datasets to approximate different data quality scenarios.

- **Near-expert**: For every task, we train a TD3 agent [11] for 1M steps and freeze its parameters. We rollout the policy with Gaussian random noise and collect 4M experience on each task.
- **Mixed**: This dataset consists of diverse data collected from various agents, including 2K near-expert trajectories for each task. Similar to ExoRL [37], we collect 10M exploratory trajectories using intrinsic reward from Proto-RL [36] for each domain. We also to use a TD3 [11] agent to maximize the sum of extrinsic reward and the Proto-RL intrinsic reward, and store its 2M experience on each task.

For more details about the above datasets, please refer to Section A. We perform both single-task and multi-task pretraining using the above datasets. The former leverages task-specific data while the latter utilizes data from all tasks within the same domain. We pretrain agents for 400k gradient steps. Specifically, for the model pretrained on the near-expert dataset, we perform zero-shot[2] evaluation of goal reaching and skill prompting, and finetuning for offline RL; for model trained on the mixed dataset, we provide the finetuning results in Section 4.3 and Section A.

Baselines

- GPT. We train an autoregressive model similar to GPT [4] which takes the past states and actions as input to predict the next state or action.
- Goal-GPT. We specifically modify GPT to Goal-GPT to evaluate its performance on goal reaching tasks. The model takes current goal and observations as input, and predicts the action to reach this goal. The model is trained using a behaviour cloning loss as [6].
- Goal-MLP. Standard behavior cloning method that conditions on the goal. The major difference between this and Goal-GPT is here we do not use the causal Transformer architecture to make the history visible.

By default, MaskDP uses a 3-layer encoder and 2-layer decoder, and the baselines based on GPT use 5 attention layers. MaskDP and all the above models are comparable with similar architecture design and size, and share the same training hyper-parameters. Details about the architecture and training of MaskDP and the above baselines can be found in Section A.

4.2 Main Results

4.2.1 Goal reaching

We consider both single and multiple goal-reaching settings. The agent is required to reach one or multiple goals from a given state, which are all sampled from the same trajectory to guarantee reachabilty within a reasonable time budget. During evaluation, the agent rolls out to reach the given goal(s) within a time budget. The evaluation dataset is also collected by the same RL agent in 3 environments with different seeds, which is unseen during pretraining. The detailed settings are:

- **single-goal reaching**: For every trajectory in the validation set, we randomly sample a start state and a future state in $T \in [15, 20)$ steps as the goal. All the methods are evaluated on the same set of 300 state-goal pairs with a given budget of $T + 3$. We set the agent to the start state[3] and report the L2 distance between the goal and the closest rollout state within this budget.
- **multi-goal reaching**: For every trajectory in the validation set, we randomly sample a start state and 5 goal states at random future timesteps from $[12, 60)$. We evaluate the same set of 100 state-goal sequences and add additional 5 timestep budgets for all the goals. Similar to single-goal reaching, We report the L2 distance between every goal and the closest rollout state before running out of its corresponding budget.

We show the zero-shot performance of MaskDP and baselines pretrained with the near-expert data (both in single-task and multi-task settings). We report L2 distance averaged over the states and goals

[2]We directly evaluate the model on some unseen state-goal pairs in the validation dataset

[3]We accomplish this by resetting the simulator.

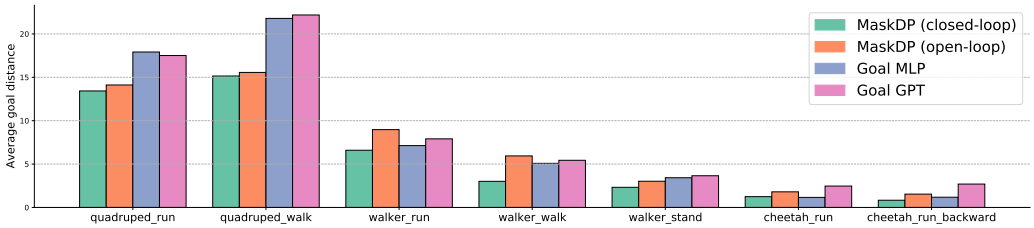

Figure 2: Single task pretraining followed by single goal reaching downstream task. MaskDP with closed-loop execution achieves the best performance on all the tasks, and get the most significant improvements in the Quadruped domain, which is higher dimensional.

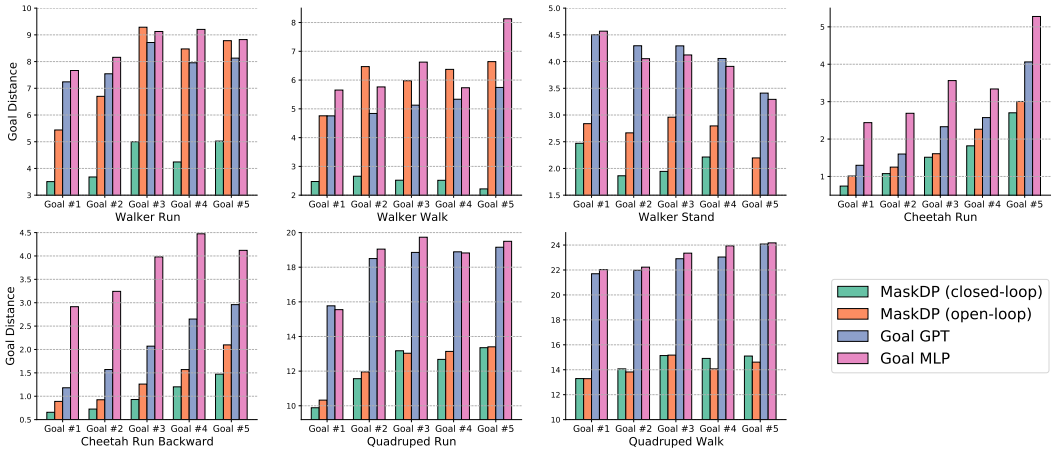

Figure 3: Single task pretraining followed by multiple goals reaching downstream task. MaskDP achieves significant improvement on all the tasks with better flexibility in sequential goal reaching.

sampled based on the above rules. Tables of run numbers and standard derivations can be found in Section C.

Figure 2 show the results of reaching single goal. The y-axis is the L2 distance (the lower the better). We observe that both MaskDP (open-loop) and MaskDP (closed-loop) outperform Goal-GPT and Goal-MLP. Despite Goal-GPT being a natural formulation for goal reaching, MaskDP reaches a lower distance to the goal. We attribute the effectiveness to learning a better understanding of the forward and inverse dynamics implicitly. We also observe that the advantages of MaskDP are even more significant in higher dimensional environments, such as Quadruped.

For the more challenging multi-goal reaching task, MaskDP has a significant advantage in flexibility: we can just provide the goals at specific time budgets with interleaved masks and get an executable plan; however, for Goal-MLP and Goal-GPT, we have to change goals at certain timesteps to fulfill future multiple goals. As shown in Figure 3, MaskDP outperforms both goal-GPT and BC by a large margin. We hypothesize that this is due to having "foresight" about future goals, which can help the agent to produce a better plan.

We can get similar conclusions from the multi-task pretrained models in Figure 4 and Figure 5, where our method consistently works well on all domains, with the most visible advantage in multi-goal reaching setup.

4.2.2 Skill Prompting

We are interested in the learned behavior of pretrained models. We use prompting, which has become popular in analyzing models ever since GPT [4]. To do so, we give the agent a short state-action segment randomly cropped from an expert trajectory, set the agent to the last state of the segment, and let the model continue to generate consecutive behaviors. We evaluate the quality of the generated sequence by comparing its obtained rewards with the rollout of a skilled expert.

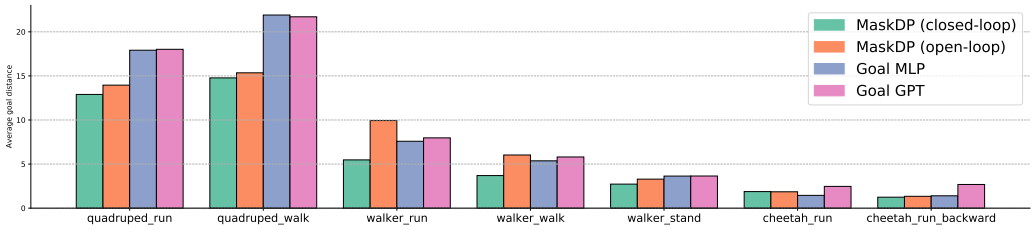

Figure 4: Multiple tasks pretraining followed by single goal reaching downstream task, where MaskDP with closed-loop execution works the best, especially in the Quadruped domain.

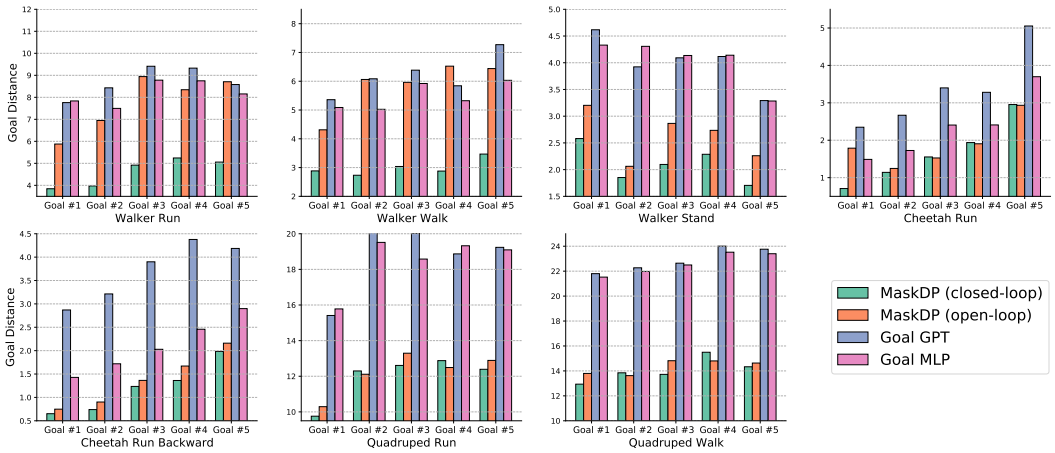

Figure 5: Multiple task pretraining followed by multiple goals reaching downstream task.

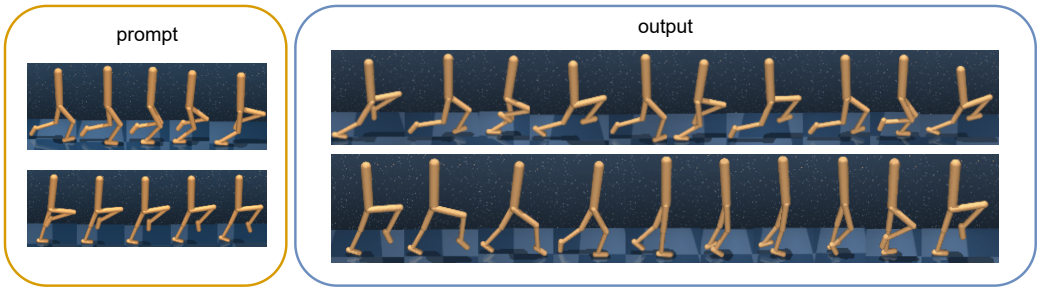

Figure 6: Qualitative results on for skill prompting in the Walker domain. Given 5 initial states, the model learns to forecast future trajectories as in the expert-level behaviour.

To be specific, we prompt the multi-task model trained with expert data and sample a 5-timestep state-action segment from $T \in [100, 900]$, where the agent can be walking/running at low or high speeds. We prompt the model with this short segment and let the model generate rollouts for 20, 40, 60 timesteps. We provide both qualitative results and quantitative results in Figure 7 and Figure 6 respectively. We can see in Figure 7, both our method and GPT can match the expert return. Although our method is not trained in an autoregressive way, it can still perform well in the sequence generation task. For the zero-shot prompting results of MaskDP/GPT trained on mixed data, see Section C.

4.2.3 Offline Reinforcement Learning

Evaluation We provide a 2M buffer of the data collected by Proto-RL [36] as in ExoRL [37] does, where the overall return of the data is quite low and thus the BC-based method cannot work well. ExoRL [37] simply shows that an offline TD3 agent works the best on diverse low-return offline data.

We can modify MaskDP to this setting by adding additional actor and critic heads on top of the pretrained encoder, and performing RL training. We evaluate the efficiency of the pretrained model

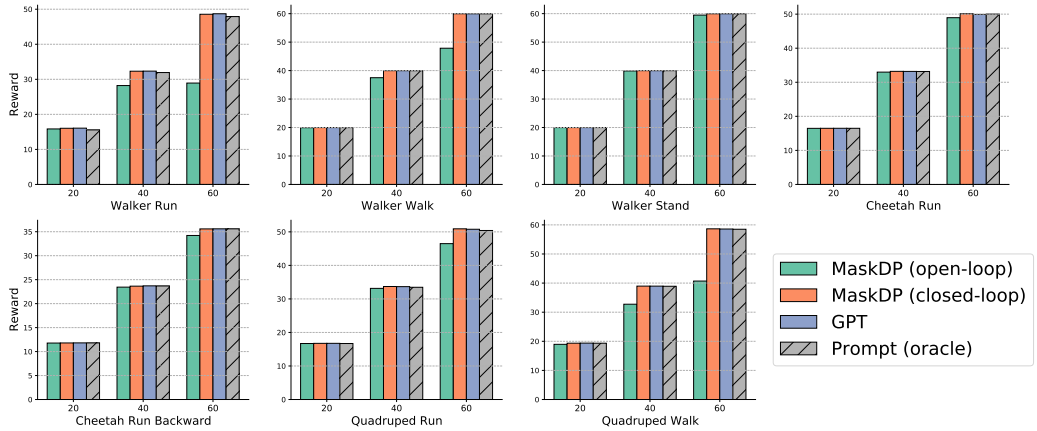

Figure 7: Quantitative results on learned behaviors using prompt. Both MaskDP and GPT can match or even slightly surpass the expert-level performance (right grey bar) in trajectory forecasting.

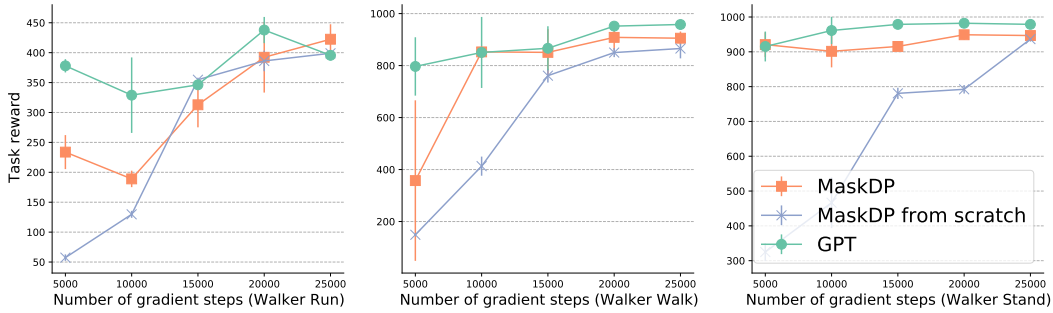

Figure 8: Offline RL results on Walker domain. The result of MaskDP matches the GPT-style pretraining performance, and are all comparable to the SoTA in ExoRL [37].

by its return after certain TD gradient steps. The results are shown in Figure 8 averaged over 3 seeds. We observe MaskDP is capable of adapting to downstream tasks quickly, outperforms training from scratch, and achieves similar results as the GPT baseline. Note that in this setting, we need to replace the bidirectional attention mask with the causal attention mask, so there is a larger gap between pretraining and downstream tasks finetuning compared with GPT, which is trained with causal masking. Note that MaskDP from scratch is almost the same as GPT from scratch (both with causal masking). From Figure 8, both MaskDP and GPT can match the best result in ExoRL [37] from their offline TD3 agent, where BC-based method cannot successfully solve this task.

4.3 Analysis

Model scalability. We also pretrain our agent and baselines on the diverse "mixed" dataset. We compare the smaller version of MaskDP and Goal-GPT[4] on the Quadruped goal reaching problem as shown in Figure 9. The x-axis is the finetuning gradient steps on the expert dataset, and y-axis is the L2 distance to the goal (the lower

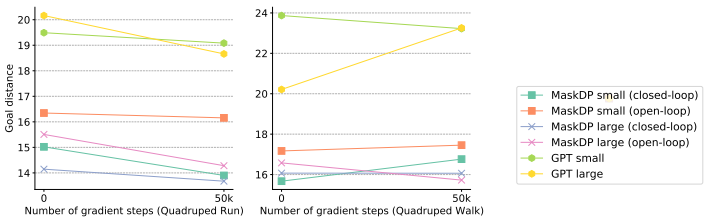

Figure 9: Model scalability on two tasks. X-axis represents number of gradient steps. With MaskDP pretraining, larger models outperform smaller models.

[4]Both use 3 attention layers.

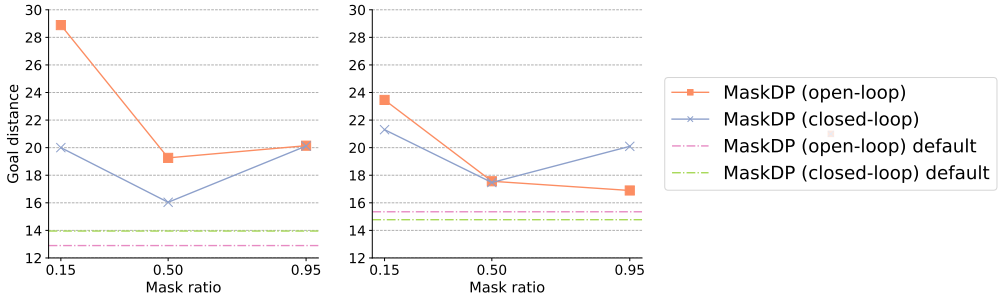

Figure 10: Mask ratio ablation. We compare our multiple ratio pretrained model with models trained with fixed ratios, where our masking strategy can achieve much better performance.

the better). We found for both zero-shot evaluation and finetuning, our model's performance improves when model size is enlarged, whereas for Goal-GPT the performance gain is not obvious.

Mask ratio ablation. Figure 10 shows the influence of the masking ratio. With a fixed mask ratio, we observe that an extremely high mask ratio (95%) generally does not work well and the typical mask ratio (15%) used in BERT seems to perform much worse than others. A middle mask ratio 50% performs reasonably well, despite still being surprisingly high, similiar to the observations in MAE. However, our mixed mask ratio strategy strictly outperforms all the above options.

Predicting unmasked tokens ablation. We also compare the model trained with mask loss vs. total loss. As shown in Figure 11, empirically we do not find mask loss has more advantages than total loss, even on the relatively clean expert dataset, it converges slower than using total loss. For the results on diverse data, please refer to Section C.

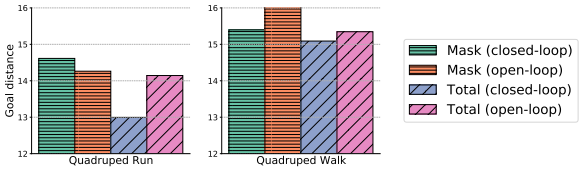

Figure 11: Masked loss and total loss ablation on 20k pretraining gradient steps. The model trained with total loss converges faster than the one trained with masked loss.

5 Conclusion

This paper presents masked decision prediction (MaskDP), a simple and scalable self-supervised method for reinforcement learning (RL) inspired by current large language and vision models. MaskDP is capable of learning scalable and generalizable agents for reinforcement learning that can learn from diverse-quality data sources and infer tasks in goal-reaching and skill-execution settings. Through our empirical study we find MaskDP models outperform past work in zero-shot goal reaching and transfer well to downstream RL tasks, performing competitively with prior pretraining and training from scratch methods.

Limitations and Future Work CV and NLP domains have shown that the true promise of masking architectures lies with their ability to ingest diverse, fully unsupervised data. We study how MaskDP performs when trained without access to expert-level data, and evaluated on unseen proprioceptive states. In the future, we can extend our method to pixel inputs, and pretrain the model to adapt to far different downstream tasks.

The architecture used in MaskDP closely resembles a model-based method, as states are predicted sequentially from actions. In this paper, we use use the predicted next actions directly as this is the simplest and fastest approach. However, it is straightforward to extend MaskDP to plan through our learned model and compare against related baselines.

Societal Impact This is an algorithm for training agents in the style of recent large-scale CV and NLP models. While we do not anticipate particular social risks from our method, as algorithms become capable of ingesting large-scale, in-the-wild data it is important to ensure the dataset does not reinforce undesirable biases or promote harmful behaviors.

6 Acknowledgements

We would like to thank Olivia Watkins, Yuqing Du, Younggyo Seo, Xingyu Lin, Danijar Hafner for insightful discussion and giving constructive comments. This work was partially supported by Office of Naval Research under grant N00014-21-1-2769 and grant N00014-22-1-2121. We would also like to thank anonymous reviewers for their helpful feedback for previous versions of our work.

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv: Arxiv-2204.14198*, 2022.

[2] G. Baldassarre and M. Mirolli. *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013.

[3] H. Bao, L. Dong, and F. Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

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

[5] M. Carroll, J. Lin, O. Paradise, R. Georgescu, M. Sun, D. Bignell, S. Milani, K. Hofmann, M. Hausknecht, A. Dragan, and S. Devlin. Towards flexible inference in sequential decision problems via bidirectional transformers. *arXiv preprint arXiv: Arxiv-2204.13326*, 2022.

[6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv: Arxiv-2106.01345*, 2021.

[7] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[8] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

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

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv: Arxiv-1802.09477*, 2018.

[12] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[13] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

[14] S. Krishnamoorthy, S. M. Mashkaria, and A. Grover. Generative pretraining for black-box optimization. *arXiv preprint arXiv:2206.10786*, 2022.

[15] M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv: Arxiv-2110.15191*, 2021.

[16] M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

[17] H. Liu and P. Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

[18] H. Liu and P. Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] T. Nguyen and A. Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

[20] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[21] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[22] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[23] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[24] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. *arXiv preprint arXiv:2206.14244*, 2022.

[26] Y. Seo, K. Lee, S. James, and P. Abbeel. Reinforcement learning with action-free pre-training from videos. *arXiv preprint arXiv:2203.13880*, 2022.

[27] Ö. Şimşek and A. G. Barto. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the 23rd international conference on Machine learning*, pages 833–840, 2006.

[28] A. Stooke, K. Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[29] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. Deepmind control suite. *arXiv preprint arXiv: Arxiv-1801.00690*, 2018.

[30] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[31] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers and distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021.

[32] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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

[34] Y. Wang, S. Joty, M. R. Lyu, I. King, C. Xiong, and S. C. Hoi. Vd-bert: A unified vision and dialog transformer with bert. *arXiv preprint arXiv:2004.13278*, 2020.

[35] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[36] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[37] D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv: Arxiv-2201.13425*, 2022.

[38] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[39] Q. Zheng, A. Zhang, and A. Grover. Online decision transformer. In *International Conference on Machine Learning*, 2022.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Will release the code in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide comprehensive details about our experiments in the Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See our experiments in section 4.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental Details

A.1 Environments and Tasks

We provide the details about the environments and tasks used in our experiments in Table 1.

Domain	Quadruped	Walker	Cheetah
Task	run walk	run walk stand	run run_backward
State dim	79	18	24
Action dim	12	6	6

Table 1: Environments and tasks from the DeepMind control suite [29] used in our experiments.

A.2 Datasets

Near-Expert Dataset We use TD3 agent [11] trained with 1M steps in the above tasks. We then freeze its parameters and rollout its policy with Gaussian noise $N(0, 0.2)$ in every dimension of the action space. For each task, we collect 4M steps of experience (4K episodes in total) using this pretrained TD3 agent.

Mixed Dataset Mixed dataset consists of the following data from various RL agents:

- Near-expert data: Same as the above near-expert dataset, but we only include 2M steps experience (2K episodes in total) for each task.
- Unsupervised data: We use an unsupervised RL algorithm, Proto-RL [36], to collect diverse unsupervised data. We train the agent for 10M steps in each domain and record all the 10M steps (10K episodes).
- Semi-supervised data: We train a TD3 agent to optimize the sum of Proto-RL [36] intrinsic reward and extrinsic reward. The agent is trained for 2M steps in each task and we record all the 2M steps (2K episodes) for each task.

A.3 Hyperparameters

We provide more details about the hyperparameters and other settings of model training and evaluation in Table 2.

A.4 Training Details

Goal-MLP Training We adapt the training of Goal-MLP to make it learn to reach goals with varying time budgets. Given a state-action sequence (s_t, a_t, s_{t+1}, ..., s_{t+m}), Goal-MLP randomly sample two states s_i and s_j as starting state and goal, (where $t \leq i < j \leq t+m$), and predicts the action a_i.

Goal-GPT Training Given a state-action sequence (s_t, a_t, s_{t+1}, ..., s_{t+m}), Goal-GPT treats $g = s_{t+m}$ as goal. Every state s_i (where $t \leq i < t+m$) is concatenated with g. Then Goal-GPT predicts the action sequence a_t, ... a_{t+m-1} from the state-goal sequence (s_t, g), ... (s_{t+m-1}, g) by passing through causal self-attention layers. In this way, all the goal-reaching baselines are pre-trained to reach goals in various timesteps.

GPT Training Given a state-action sequence (s_t, a_t, s_{t+1}, ..., s_{t+m}), GPT predicts the next token (state or action) conditions on previous token sequence, $i.e.$, predicting s_j ($j > t$) from s_t, a_t, ... s_{j-1}, a_{j-1}.

A.5 Compute Resources

MaskDP is designed to be accessible to the RL research community. The whole pipeline, including data collection, pretraining, and finetuning, only requires a single GPU. All experiments were run on

MaskDP	**Value**
# Context length	64
# Encoder layer	3
# Decoder layer	2
# Attention head	4
# Hidden dimension	256
Mask ratio	[0.15, 0.35, 0.55, 0.75, 0.95]
GPT/Goal-GPT	**Value**
# Context length	64
# Attention layer	5
# Attention head	4
# Hidden dimension	256
Goal-MLP	**Value**
# Context length	64
# Linear layer	5
# Hidden dimension	1024
Training	**Value**
Optimizer	Adam
(β_1, β_2)	$(.9, .999)$
Learning rate	$1e^{-4}$
Batch size	384
# Gradient step	400000
Evaluation	**Value**
# seed	3
# Goals (single-goal reaching) per seed	300
# Goals (multi-goal reaching) per seed	100×5
Prompt context length	5
Discount (for RL)	0.99
Replay buffer size (for RL)	$2M$

Table 2: Hyperparameters used for model training and evaluation.

GPU clusters with 8 NVIDIA TITAN Xp. The pretraining takes 6-8 hours for 400k gradient steps on the collected datasets using a single GPU.

B Additional Experiments

B.1 Dataset Quality

MaskDP has no assumption about the pretraining dataset. To show it doesn't rely on the expert data, we reconstruct another highly diverse dataset called **mixed-v2**, which contains:

- unsupervised data: we train a TD3 [11] agent to maximize Proto-RL [36] intrinsic reward, and store its 10M replay buffer on each domain.
- semi-supervised data: we train a TD3 agent to maximize the sum of extrinsic reward and the Proto-RL intrinsic reward, and store its 2M replay buffer on each task.
- supervised data: we train a TD3 agent to maximize extrinsic reward and store its 2M replay buffer on each task.

The **mixed-v2** dataset is more diverse, as it replaces the near-expert data with TD3 training samples, which are more suboptimal and noisy. After pretraining using **mixed-v2**, we evaluate its performance on unseen state-goal pairs from **near-expert** dataset (dataset in the main paper). So the pretraining and evaluation datasets are in different distributions.

In Figure 12 and Figure 13, we find our model consistently outperforms baselines on all the domains. Compared with Figure 4 and Figure 5, it has more advantages when dataset is noisy, as BC-based methods highly rely on the dataset quality.

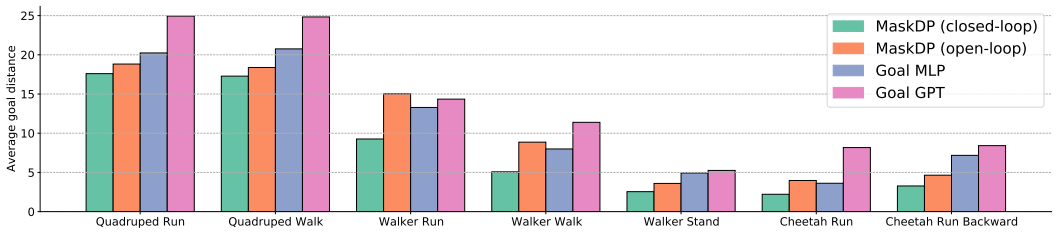

Figure 12: The single goal reaching results on **near-expert** goal reaching, after pretraining MaskDP on **mixed-v2** dataset.

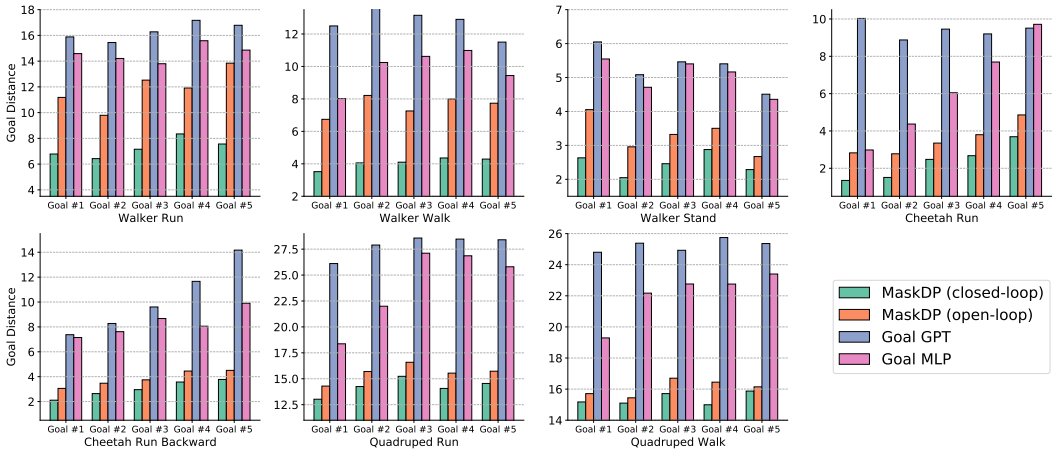

Figure 13: The multiple goal reaching results on **near-expert** goal reaching, after pretraining MaskDP on **mixed-v2** dataset.

B.2 Multi-goal Reaching with foresight

We add ablation about whether to provide multiple future goals to the agent for multi-goal reaching. In contrast, We can also give the agent an individual goal at one time, and switch to a different goal when the budget is exhausted.

B.3 Trajectory Length

As in [12, 31], we also use sinusoidal positional embedding and perform linear interpolation when the trajectory is longer than the training time. Figure 15 shows the results when we execute the agent for 60, 90, and 120 steps with 5 context tokens, where the training trajectory length is 64. We found on most environments, closed-loop MaskDP can achieve similar performance with GPT and the expert return (the gray bar), except for Cheetah tasks. For longer trajectories, the mask ratio can be extremely low at the beginning, which can cause some bad initial behavior. Meanwhile, GPT can perform stably well with interpolated positional embedding.

B.4 Additional RL results

We use the supervised data in **mixed-v2** to finetune RL on quadruped and cheetah domains. We keep the pre-trained encoder and train additional actor and critic heads on top of the encoder. Figure 16 shows the performance on Quadruped and Cheetah domain, averaged over 2 seeds.

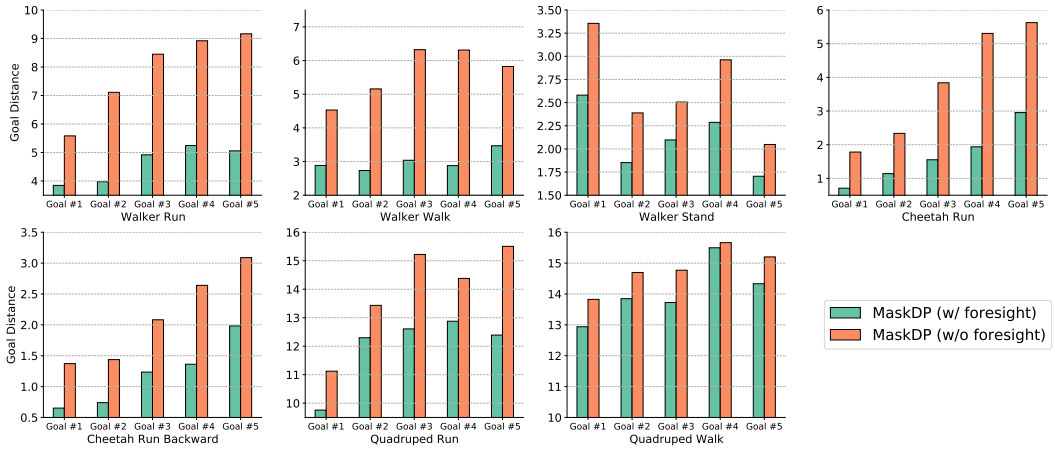

Figure 14: We test the closed-loop performance of MaskDP to understand whether the visibility of future goals can improve the performance. We found that on all the domains, MaskDP with foresight performs better.

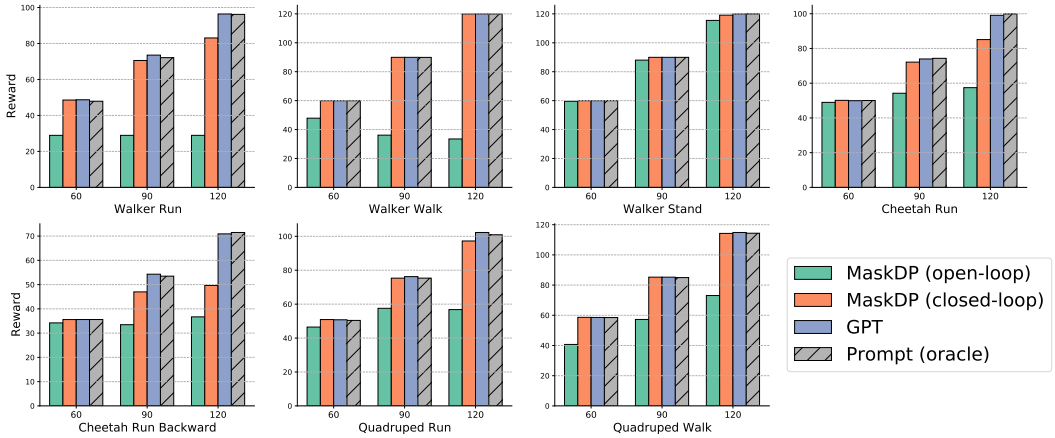

Figure 15: Skill prompting performance for longer rollouts.

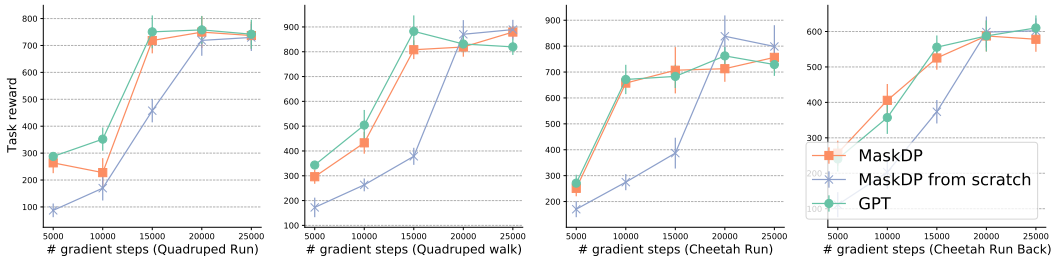

Figure 16: RL finetuning results on Quadruped and Cheetah domain. MaskDP and GPT are more efficient than being trained from scratch.

B.5 Jaco results

We also add a Jaco arm reaching task in the robotics domain. The training and evaluation both follow Section B.1 As shown in Figure 17, MaskDP still outperforms baselines on this task.

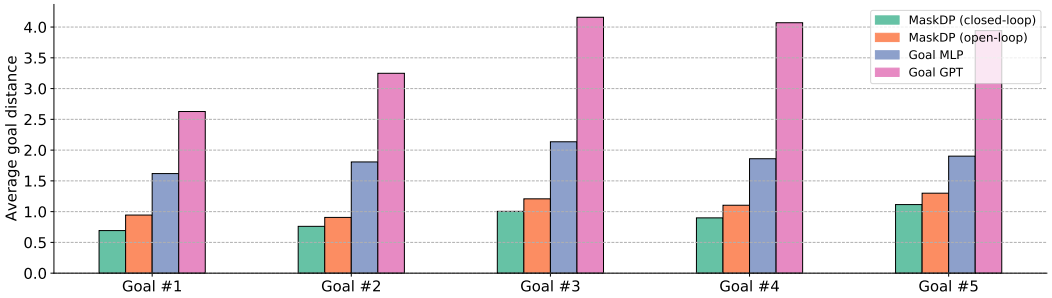

Figure 17: Multi-goal reaching results on Jaco reaching task. MaskDP outperforms baselines on this domain.

C Full Experimental Results

C.1 Single-goal Reaching Results

We provide the single-goal reaching results in Table 3 and Table 4 for single-task and multi-task respectively.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	17.832±0.321	18.313±0.171	13.753±0.255	12.912±0.018
	walk	22.965±0.077	23.051±0.055	15.456±0.176	15.116±0.396
Walker	run	8.15±0.080	9.197±0.012	9.565±0.157	7.789±0.220
	walk	5.111±0.118	6.029±0.415	5.822±0.281	2.63±0.158
	stand	3.979±0.293	4.108±0.317	3.242±0.1527	2.393±0.076
Cheetah	run	1.377±0.037	2.878±0.057	2.234±0.145	1.385±0.122
	run backward	1.447±0.136	3.011±0.095	1.752±0.052	0.866±0.065

Table 3: single-task single-goal reaching results.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	17.825±0.218	18.224±0.551	13.214±0.230	12.926±0.382
	walk	22.977±0.484	23.361±0.201	14.892±0.140	15.428±0.203
Walker	run	8.600±0.095	8.977±0.181	10.242±0.149	6.233±0.149
	walk	5.550±0.170	6.083±0.392	6.584±0.487	4.042±0.148
	stand	4.096±0.015	4.137±0.408	3.422±0.161	2.248±0.026
Cheetah	run	1.634±0.042	2.953±0.085	1.924±0.099	1.939±0.125
	run backward	1.694±0.061	2.980±0.076	1.378±0.055	1.395±0.011

Table 4: Multi-task single-goal reaching results.

C.2 Multi-goal Reaching Results

We provide multi-goal reaching results for multi-task pretrained models in Table 5, Table 6, Table 7, Table 8 and Table 9.

C.3 Finetuning Results on Model Scalability

We pretrain MaskDP using the diverse multi-task mixed dataset, and finetune it using near-expert dataset on each task. In addition to the results in Figure 9 on the Quadruped domain, we also provide

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	15.644±0.193	15.925±0.726	10.396±0.152	10.213±0.644
	walk	21.963±0.241	22.114±0.445	13.413±0.545	12.99±0.073
Walker	run	7.562±0.385	7.588±0.243	5.981±0.006	4.032±0.265
	walk	5.238±0.212	5.481±0.172	4.256±0.080	2.721±0.213
	stand	4.295±0.050	4.483±0.189	2.998±0.271	2.293±0.426
Cheetah	run	1.381±0.151	2.342±0.009	0.995±0.132	0.738±0.041
	run backward	1.356±0.102	2.829±0.058	0.811±0.089	0.647±0.007

Table 5: Distance to the first goal in multi-task multi-goal reaching.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	18.183±1.883	17.915±0.649	11.44±1.065	11.736±0.796
	walk	22.628±1.006	23.225±1.325	13.986±0.515	14.487±0.937
Walker	run	8.161±0.939	8.98±0.775	7.165±0.305	4.398±0.599
	walk	5.576±0.778	6.458±0.509	6.435±0.507	2.886±0.217
	stand	4.389±0.119	4.013±0.131	2.566±0.508	2.045±0.282
Cheetah	run	1.814±0.126	2.75±0.115	1.207±0.056	1.163±0.032
	run backward	1.802±0.118	3.341±0.180	0.917±0.022	0.853±0.161

Table 6: Distance to the second goal in multi-task multi-goal reaching.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	18.377±0.324	18.911±1.58	12.334±1.26	12.085±0.710
	walk	22.787±0.269	23.907±0.946	14.7±0.166	14.069±0.498
Walker	run	9.05±0.3809	9.178±0.278	9.053±0.157	5.045±0.179
	walk	6.127±0.268	6.603±0.489	5.895±0.097	3.113±0.106
	stand	4.227±0.193	4.093±0.225	2.878±0.018	2.012±0.115
Cheetah	run	2.329±0.108	3.216±0.259	1.456±0.085	1.519±0.048
	run backward	2.11±0.114	3.807±0.131	1.291±0.102	1.216±0.026

Table 7: Distance to the third goal in multi-task multi-goal reaching.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	19.564±0.142	19.227±0.156	12.92±0.52	12.89±0.030
	walk	23.475±0.342	24.119±0.165	14.235±0.800	14.607±0.698
Walker	run	8.374±0.536	8.926±0.582	7.787±0.789	4.713±0.751
	walk	5.548±0.282	5.988±0.193	6.248±0.402	2.884±0.007
	stand	4.195±0.086	4.07±0.103	2.704±0.043	2.144±0.210
Cheetah	run	2.501±0.130	3.537±0.361	1.929±0.034	1.971±0.047
	run backward	2.491±0.047	4.145±0.330	1.825±0.217	1.48±0.166

Table 8: Distance to the fourth goal in multi-task multi-goal reaching.

Domain	Task	Goal-MLP	Goal-GPT	Ours (open-loop)	Ours (closed-loop)
Quadruped	run	18.749±0.796	19.083±1.376	13.057±0.202	12.162±0.084
	walk	23.772±0.668	24.152±1.002	15.275±0.910	15.24±1.264
Walker	run	8.563±0.612	8.567±0.196	8.955±0.352	5.338±0.392
	walk	6.876±1.103	8.334±1.613	7.231±1.101	3.664±0.276
	stand	3.93±0.796	3.775±0.648	2.539±0.394	2.009±0.414
Cheetah	run	3.375±0.456	4.623±0.609	2.769±0.232	2.716±0.335
	run backward	2.737±0.229	4.07±0.164	2.244±0.118	1.981±0.002

Table 9: Distance to the fifth goal in multi-task multi-goal reaching.

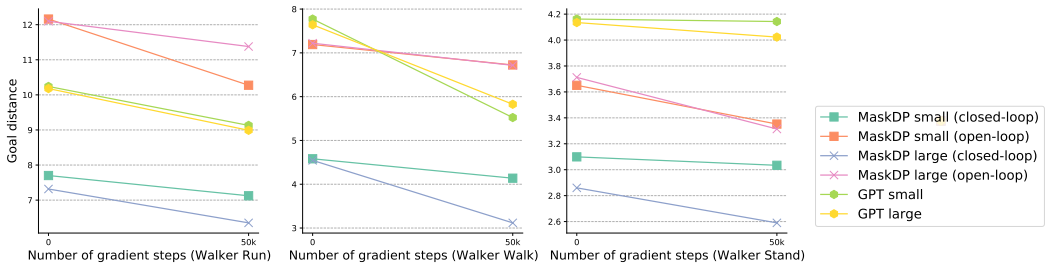

Figure 18: Model scalability on walker domain. X-axis represents number of gradient steps. With MaskDP pre-training, larger models outperform smaller models across all Walker tasks. In contrast, Goal-GPT does not have such properties.

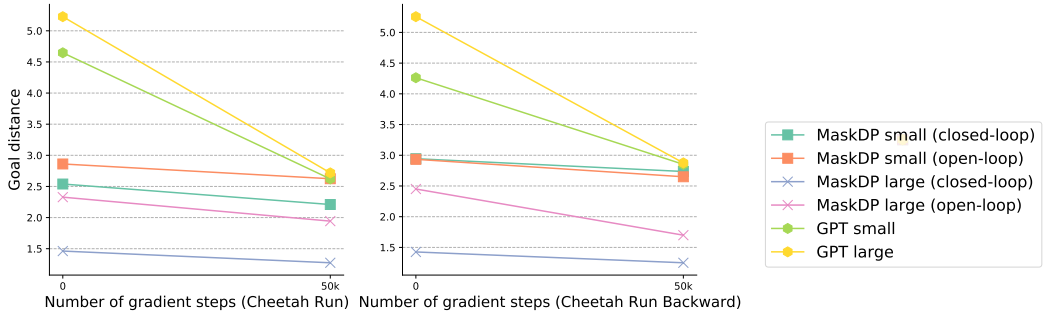

Figure 19: Model scalability on cheetah domain. X-axis represents number of gradient steps. With MaskDP pre-training, larger models outperform smaller models across all Walker tasks. In contrast, Goal-GPT does not have such properties.

results on the other two domains in Figure 18 and Figure 19. Here "small" represents a model with 3 attention layers, while "large" represents 5 attention layers.

We can see the large model with closed-loop evaluation always performs the best, while for Goal-GPT the results are much worse, and the gain from the large model is not significant.