Exploration of Sequence Modeling for Reinforcement Learning CPSC 533V 2023W2 Project Report

Juntai Cao[♠]
Student #: 50171404
jtcao7@cs.ubc.ca

Yuwei Yin[♠]
Student #: 36211928
yuweiyin@cs.ubc.ca

Xiang Zhang[♠]
Student #: 93035806
wyattz23@cs.ubc.ca

Department of Computer Science, University of British Columbia
2366 Main Mall, Vancouver, BC V6T 1Z4, Canada

Abstract

Reinforcement learning (RL) offers a learning framework tailored to decision-making and control mechanisms. Offline reinforcement learning algorithms, constituting an important branch of RL, present significant potential in harnessing large datasets to train robust decision-making policies. In this project, we explore sequence modeling for RL, focusing on the Decision Transformer, a representative offline RL model, and the utilization of offline datasets. The Decision Transformer predicts the most likely action by conditioning on the previous rewards, states, and actions, which are modeled as a sequence in the offline trajectories. We first re-implement the whole framework and then conduct extensive experiments to investigate the model performance on different model architectures and sequence tokenization settings. Experimental results and analysis deepen the understanding of the sequence modeling approach for RL tasks. In addition, we propose to train a soft actor-critic policy on the MuJoCo environment to obtain offline datasets of a newer version. Although we did not train an expert-level policy, this attempt and the corresponding discussions bring valuable insights for further research and future work. Our code and data are publicly available.[1]

1 Introduction

Searching and decision-making under uncertainty is a pivotal aspect of artificial intelligence (AI), with profound implications across different domains, especially robotics. The ability to make informed decisions in the face of incomplete or ambiguous information enables the development of robust and adaptive systems. Reinforcement learning (RL) (Sutton, 1988; Watkins and Dayan, 1992; Sutton et al., 1999; Konda and Tsitsiklis, 1999) is a powerful AI paradigm to make sequential decisions by learning from the consequences of actions. In RL (Sutton and Barto, 2018), an agent learns to achieve a goal in an uncertain, potentially complex environment by trial and error, receiving feedback through rewards or penalties. This approach allows the agent to develop a strategy, or policy, for selecting actions that maximize the cumulative reward over time.

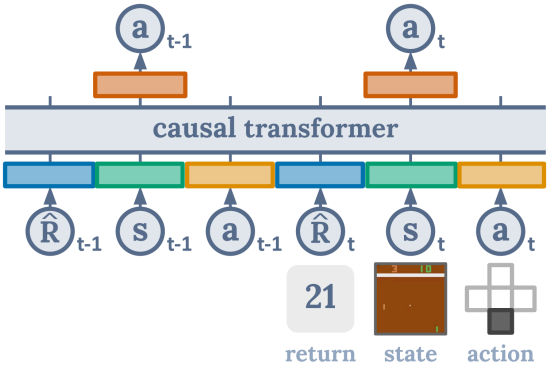

Figure 1: The overview of Decision Transformer (Chen et al., 2021).

As deep learning (DL) (Goodfellow et al., 2016) prospers in various AI domain, deep reinforcement learning exhibits excellent ability in agent control via value-based (Mnih et al., 2013, 2015) and policy-based (Silver et al., 2014; Lillicrap et al., 2016; Henderson et al., 2018) methods. Natural language processing (NLP), another AI field facilitated by deep learning, is also a sequence modeling task (Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Chung et al., 2014) and Markov decision process. Thus, ***applying the powerful sequence models in NLP for RL tasks is intuitively promising***. Transformers (Vaswani et al., 2017), with their ability to process sequential data and capture long-range dependencies, have revolutionized the way machines understand and generate natural language (Devlin et al., 2019; Radford et al., 2018).

Following this line of research, previous works (Chen et al., 2021; Janner et al., 2021; Radosavovic et al., 2024) leverage the Transformer

♠ Authors contributed equally and listed alphabetically.
[1] https://github.com/YuweiYin/UBC_CPSC_533V

architecture to model reinforcement learning-based decision-making processes. This paradigm shift allows us to directly apply the powerful capabilities of Transformer to RL problems, enabling the model to learn optimal policies from sequences of states, actions, and rewards.

There is a substantial amount of research applying transformers to offline locomotion prediction. While many of these methods predominantly leverage decoder-only Transformers, like GPT (Radford et al., 2018, 2019), as the backbone model, the effect of different model architectures, such as the depth and width of the neural networks, is still underexplored. Furthermore, although novel tokenization methods have been introduced, there is still a deficiency in comparative analyses regarding the influence of different tokenization techniques on model performance.

In this work, the main focus is to ***explore sequence modeling for reinforcement learning by framing the RL task as a sequence modeling NLP task***. Therefore, we propose to delve deep into these issues through a series of experiments. The contributions of this project are as follows:

- We review and explore sequence modeling for offline reinforcement learning in depth. Our experimental results, analysis, and findings provide insights for further research.
- We re-implement the Decision Transformer code base and conduct extensive experiments to explore its architecture and tokenization. Our results are reproducible and the code and data are publicly available.
- We train an RL policy in the MuJoCo environment to collect "v4" offline trajectories and propose to utilize the datasets for transfer learning and other future works.

2 Related Work

2.1 Offline Reinforcement Learning

Offline Reinforcement Learning, often called batch reinforcement learning, refers to training RL agents from pre-collected datasets, without real-time interactions or online explorations (Levine et al., 2020). Offline RL methods can be broadly classified into two main categories: model-based and model-free. Model-based methods focus on constructing a dynamics model from pre-collected datasets. This learned model is then used for future planning and decision-making (Yu et al., 2021). In contrast, model-free methods directly learn a policy or value function from these datasets without the need to simulate or model the environment's dynamics (Pong et al., 2018; Prudencio et al., 2023). Within the scope of our project, the sequence modelling transformer models we are exploring fall under the category of model-free algorithms.

2.2 Sequence Modeling for RL

In light of the success and growing popularity of Transformer (Vaswani et al., 2017) in natural language processing and computer vision (Dosovitskiy et al., 2021) tasks in recent years, there has been an increased interest in applying transformer models to offline reinforcement learning scenarios. Two pioneering work in this direction are Decision Transformer (Chen et al., 2021) and Trajectory Transformer (Janner et al., 2021). Although both of them mirror the causal Transformer structure of GPT (Radford et al., 2018), they exhibit some differences during inference time, which is discussed in detail in section 3.1.

2.3 Real World Control by Next Token Prediction

More recently, Radosavovic et al. (2024) re-framed the challenge of real-world humanoid control as a next token prediction problem, introducing the concept of aligned prediction to adeptly manage missing modalities. By merging the embeddings of observations and actions and treating them as a singular token, they use a causal Transformer to predict the next observation-action pairs. This approach allows for a straightforward handling of missing input modalities through the use of masks, whereby the model can simply disregard the loss associated with predictions corresponding to these masked parts of the input.

Inspired by the versatility of Large Language Models (LLMs) in handling diverse tasks, Zhang et al. (2024) investigate the use of the Transformer architecture to develop a foundational controller applicable to general control applications. They utilize a language model pre-trained on next-token prediction tasks and fine-tune it on offline control datasets using Low-Rank Adaptation (LoRA) (Hu et al., 2021). Their results demonstrate that the Transformer model can identify parameter-agnostic structures intrinsic to control tasks, showing remarkable zero-shot generalization capabilities for entirely new tasks and quickly exceeding expert

performance levels with a minimal amount of demonstration data.

3 Method

3.1 Sequence Modeling for RL

Decision Transformer. Decision Transformer (Chen et al. (2021)) leverages the Transformer model (Vaswani et al., 2017) to address reinforcement learning tasks by framing them as sequence modeling problems. Unlike traditional RL approaches (Sutton and Barto, 2018) that explicitly model the environment's dynamics or optimize a policy, the Decision Transformer directly generates actions based on past experiences, treating the problem as a supervised learning task. The Decision Transformer utilizes the reward-to-go $\hat{R}_t = \sum_{t'=t}^{T} r'_t$, rather than the immediate reward or total accumulated reward, enabling the model to generate actions based on future returns instead of rewards already obtained. Thereby the trajectory representation is formulated as in equation 3.1. During training step, the three modalities (reward-to-go, state, action) of the last K timesteps are stacked as a single token embedding $(\hat{R}_t, s_t, a_t)$ and processed by GPT-2 (Radford et al. (2019)) to predict future action.

$$\tau = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_T, s_T, a_T) \tag{3.1}$$

Trajectory Transformer. In discrete states and actions, just like the Decision Transformer, Trajectory Transformer (Janner et al. (2021)) process the trajectory data as a sequence input, but use the immediate reward r_t instead of reward-to-go. The representation of trajectory in the trajectory transformer is shown in Equation 3.2.

$$\tau = (s_1, a_1, r_1, \ldots, s_T, a_T, r_T) \tag{3.2}$$

However, in the event of continuous states and actions, Trajectory Transformer discretize each dimension independently. Assuming N-dimensional states and M-dimensional actions, this will turn the representation of the trajectory τ into sequence of length $T = (N + M + 1)$ as in Equation 3.3.

$$\tau = (\ldots, s_t^1 \ldots, s_t^N, a_t^1, \ldots, a_t^M, r_t, \ldots) \tag{3.3}$$

Moreover, unlike the Decision Transformer which is focused on conditioning on reward-to-go, Trajectory Transformer generates future actions with beam search during inference time. Since using beam search as a reward-maximizing procedure may result in policies that prioritize short-term gains at the expense of potentially higher long-term returns in offline setting, the discounted reward-to-go term $\hat{R}_t = \sum_{t'=t}^{T} \gamma^{t'-t} r'_t$ is augmented to each transition in the training trajectories. The full trajectory representation used in training is shown in Equation 3.4.

$$\tau = (\ldots, s_t^1 \ldots, s_t^N, a_t^1, \ldots, a_t^M, r_t, \hat{R}_t \ldots) \tag{3.4}$$

Given the conceptual similarities between the ideas of Decision Transformer and Trajectory Transformer, our analysis will focus on the Decision Transformer to explore sequence modeling in offline RL.

3.2 Training RL Policies in MuJoCo

In order to obtain our own offline data, we train a RL policy from scratch in the MuJoCo environment. Thus, we explore the methods of Actor Critic (Konda and Tsitsiklis, 1999) and its modified version–Soft Actor Critic (Haarnoja et al., 2018).

Actor-Critic. In the actor-critic framework, the actor is responsible for selecting actions based on the current policy, while the critic evaluates these actions by computing the value function of the state-action pair. This separation helps in stabilizing the training process compared to methods that use value functions alone. The critic's assessment informs adjustments to the actor's policy, ideally leading to more optimal actions over time.

Training involves two networks: one for the actor that outputs the policy (action probabilities), and one for the critic that outputs a value score for the state-action pairs. The networks are updated through gradients calculated from the temporal difference (TD) (Sutton, 1988) error, provided by the critic.

Soft Actor-Critic. Soft Actor-Critic, an extension of the AC framework, incorporates principles from entropy maximization, encouraging the actor not only to maximize expected reward but also to maintain a diverse range of actions (maximizing entropy).

4 Experimental Setup

In this section, we introduce the experimental setup of the project in detail.

Env	# Action	# State
Half Cheetah	6	17
Hopper	3	11
Walker2D	6	17

Table 1: The statistics of the target MuJoCo environments.

Env	Level	Min	Max	Mean	Std
Half Cheetah	Random	-524.0	-85.4	-288.8	80.4
	Medium	-310.2	5309.4	4770.3	355.8
	Expert	2045.8	11252.0	10656.4	441.7
Hopper	Random	2.9	292.6	18.4	17.5
	Medium	315.9	3222.4	1422.1	379.0
	Expert	1645.3	3759.1	3511.4	328.6
Walker2D	Random	-17.0	75.0	1.9	5.8
	Medium	-6.6	4226.9	2852.1	1095.4
	Expert	763.4	5011.7	4920.5	136.4

Table 2: The statistics of the "v2" offline datasets.

4.1 Environment and Datasets

Here, we introduce the reinforcement environment, offline datasets, and evaluation metrics.

MuJoCo Environment. We experiment on three MuJoCo tasks (Todorov et al., 2012)[2], i.e., Half Cheetah (Wawrzynski, 2007; Wawrzyński, 2009), Hopper (Erez et al., 2011), and Walker2D. The statistics of the target MuJoCo environments is shown in Table 1, including the number of actions/states. The action value range of every environment is (-1.0, 1.0) and the state value range of every environment is $(-\infty, \infty)$. The action data type of every environment is `float32` and the state data type of every environment is `float64`. For simplicity, we use "state" and "observation" interchangeably in this paper.

V2 Offline Datasets. We use the offline dataset from D4RL (Fu et al., 2020)[3]. The trajectories are collected by observing AI agents (RL policies) interacting with above three RL environments ("v2" version). Each dataset has three types (levels) of subsets ("random", "medium", and "expert") denoting different trajectory qualities. Table 1 shows the statistics of the statistics of the "v2" offline datasets. Each trajectory has a max length of 1000. For every data point in the trajectory, there is a reward value, a list of states, a list of actions, and a boolean variable to indicate the end of the trajectory available.

Evaluation Metrics. The performance of RL policies is measured by the reward $r \in \mathbb{R}$ from the RL environment, i.e., the higher the reward, the better. The environment is based on the Gymnasium [4] library, which is rooted from OpenAI Gym (Brockman et al., 2016).

4.2 Experiment 1: Baseline Model

We re-implement an updated version of the code base for the decision-making Transformer based on the original implementation of the Decision Transformer[5], which has obsolete code pieces and package dependencies that results in bugs.

Training Details. The hyper-parameters of our baseline model follow the original configuration: 3 Transformer layers with 1 attention head and the hidden state dimension is 128. For all the Decision Transformer experiments, we train the model for 10 epochs with a default batch size of 64 and a learning rate of 1e-4.

4.3 Experiment 2: Architecture Exploration

Decision Transformer utilizes GPT-2 (Radford et al., 2019) as the backbone, which is characterized by a decoder-only structure and is trained through a next-token prediction task. We explore various architecture settings of the Transformer (Vaswani et al., 2017). Specifically, we try different hidden state dimensions, numbers of layers, numbers of attention heads, and training batch sizes.

4.4 Experiment 3: Tokenization Exploration

The methods mentioned in § 2 differ mainly in the tokenization approach (i.e., the permutation of rewards, observation/state, and actions) in the trajectory of the Transformer encoder.

We first examine whether the predictive capabilities of the Decision Transformer are robust to variations in the stacking sequences of its input components. Specifically, the model integrates the reward-to-go, state, and action at each timestep into a cohesive unit. Given that the model's predictions of the next block (state, action and reward-to-go) are based on the entire block from current timestep, we expect different input configurations would not compromise the model's predictive accuracy in this setting.

[2] https://gymnasium.farama.org/environments/mujoco/

[3] https://github.com/Farama-Foundation/D4RL

[4] https://github.com/Farama-Foundation/Gymnasium

[5] https://github.com/kzl/decision-transformer

Figure 2: The training loss curves of Decision Transformer experiments.

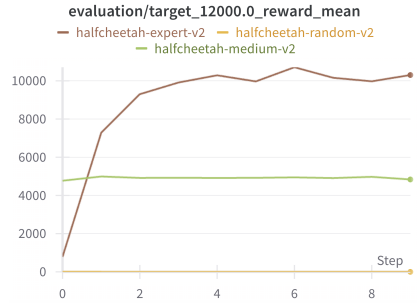

Figure 3: The evaluation rewards of Decision Transformer trained on different levels of offline datasets in the Half Cheetah environment.

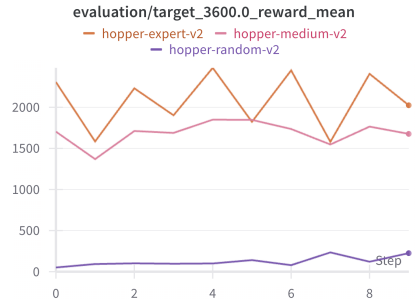

Figure 4: The evaluation rewards of Decision Transformer trained on different levels of offline datasets in the Hopper environment.

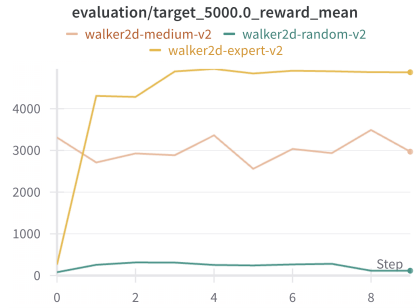

Figure 5: The evaluation rewards of Decision Transformer trained on different levels of offline datasets in the Walker2D environment.

Secondly, the original implementation relies only on current action to sample the new state, which may result in sub-optimal policies due to lack of the current state information. Thus, we propose to utilize both state and action inputs for predicting the reward-to-go and the subsequent state. We expect this input strategy to enhance the model by incorporating state and action embeddings during prediction.

4.5 Experiment 4: V4 Dataset

The readily-available trajectories in D4RL (Fu et al., 2020) are collected in "v2" version of the MuJoCo environment. We aim to obtain a "v4" offline datasets by training RL policies on the MuJoCo-V4 environment. Specifically, we train a RL policy from scratch using the actor-critic(AC) (Konda and Tsitsiklis, 1999) and soft actor-critic (SAC) approach, as introduced in § 3.2.

5 Results and Analysis

In this section, we present the experimental results and corresponding analysis. The code and data are publicly available.

5.1 Experiment 1: Baseline Model

As described in § 4.2, we re-implement Decision Transformer, train the model on the "v2" offline datasets, and test the performance on "v4" MuJoCo environment. As the training losses in Figure 2, all the Decision Transformer experiments run successfully.

As described in § 4.1, the MuJoCo-V2 datasets present random, medium, and expert level "v2" datasets with different qualities (rewards). We train the Decision Transformer model on each level of "v2" datasets and evaluate the performance on all three MuJoCo-V4 environments. The evaluation rewards of Decision Transformer trained on different levels of offline datasets in the Half Cheetah, Hopper, and Walker2D environments are shown in Figure 3, Figure 4, and Figure 5, respectively.

It is clear that the quality of the expert-level dataset is the best and the random-level dataset is the worst. Decision Transformer successfully achieves the V4-env performance (reward) by training on the "v2" offline dataset.

5.2 Experiment 2: Architecture Exploration

As described in § 4.3, we explore various architecture settings of the Transformer by experimenting on different hidden state dimensions, numbers of layers, numbers of attention heads, and training batch sizes. Here, we use the Half Cheetah environment and the medium-level "v2" dataset for

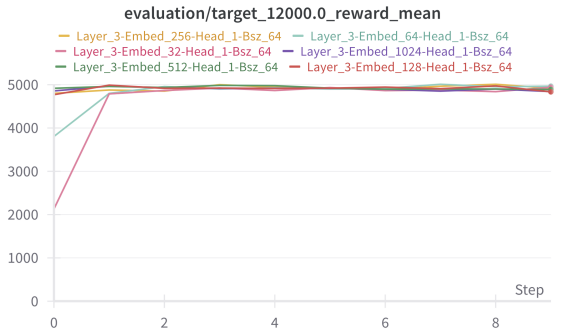

Figure 6: The evaluation rewards of Decision Transformer with different hidden state dimensions (default: 128) trained on medium-level offline datasets in the Half Cheetah environment.

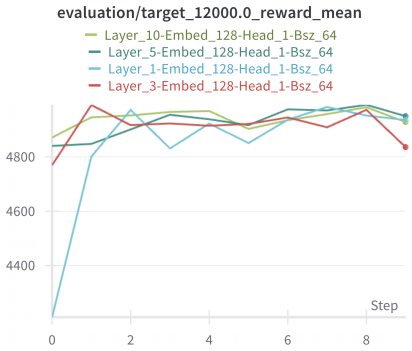

Figure 7: The evaluation rewards of Decision Transformer with different numbers of layers (default: 3) trained on medium-level offline datasets in the Half Cheetah environment.

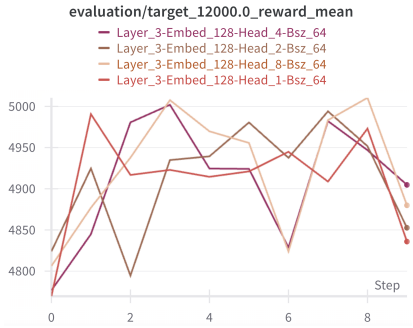

Figure 8: The evaluation rewards of Decision Transformer with different numbers of attention heads (default: 1) trained on medium-level offline datasets in the Half Cheetah environment.

training.

Hidden state dimensions. Figure 6 shows the evaluation rewards of Decision Transformer with different hidden state dimensions (default: 128) trained on medium-level offline datasets in the Half Cheetah environment.

As depicted in the figure, models with different hidden state sizes ultimately converge to a similar mean reward of approximately 6000. However, it is notable that models with larger hidden state sizes converge faster than those with smaller sizes. This increased convergence speed is particularly evident when the hidden state dimension is smaller than 128, suggesting that dimensions greater than 128 may be redundant. Consequently, a dimension of 128, the default setting, appears to be the optimal value for the hidden state.

Numbers of layers. Figure 7 shows the evaluation rewards of Decision Transformer with different numbers of layers (default: 3) trained on medium-level offline datasets in the Half Cheetah environment.

Similarly, we observe that variations in the number of layers do not impact the final convergence of the model. All configurations eventually reach a mean reward of approximately 6000, with negligible differences between them. Notably, a model with a single layer converges much slower compared to models with multiple layers. However, the benefits of increased convergence speed appear to be marginal to nonexistent when the number of layers exceeds three. Therefore, the default setting of three layers is deemed optimal.

Numbers of attention heads. Figure 8 shows the evaluation rewards of Decision Transformer with different numbers of attention heads (default: 1) trained on medium-level offline datasets in the Half Cheetah environment.

As illustrated in the figure, models with varying numbers of attention heads display no clear performance pattern, with results fluctuating significantly between 4000 and 5000. Given the lack of consistent evidence that a specific number of heads yields substantial training benefits, we have chosen to disregard differences in the number of attention heads and stick our experiments with default head value.

Training batch sizes. Figure 9 shows the evaluation rewards of Decision Transformer with different training batch sizes (default: 64) trained on medium-level offline datasets in the Half Cheetah environment.

Not surprisingly, a lower batch size results in slower convergence to peak performance, although, as illustrated in the figure, all batch sizes eventually converge to a similar mean reward. Therefore, we select 64 as the optimal batch size to balance the training efficiency with resource utilization effectively.

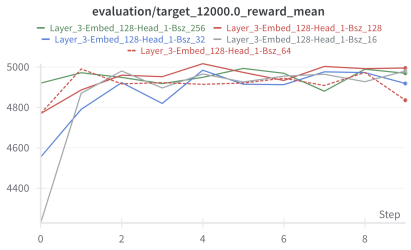

Figure 9: The evaluation rewards of Decision Transformer with different training batch sizes (default: 64) trained on medium-level offline datasets in the Half Cheetah environment.

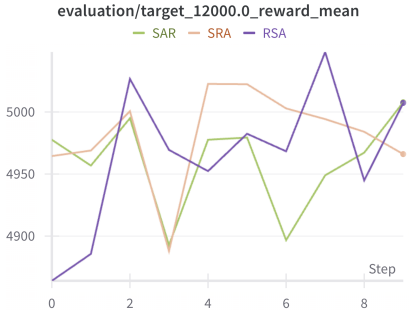

Figure 10: The evaluation rewards of Decision Transformer with different stacking sequences (baseline: RSA) trained on medium-level offline datasets in the Half Cheetah environment.

5.3 Experiment 3: Tokenization Exploration

As described in § 4.4, we explore different tokenization approaches in the trajectory of the Transformer encoder. Similar to Experiment 2 (§ 5.2), we use the Half Cheetah environment and the medium-level "v2" dataset for training.

Robustness to Stacking Sequences in Trajectory. Figure 10 shows the evaluation rewards of Decision Transformer with different stacking sequences. Our analysis indicates that all stacking sequences yield approximately the same mean rewards. Consistent with our expectations, the structure of Decision Transformer demonstrates robustness to variations in the stacking sequences of its inputs. This resilience allows the model to adapt effectively to different input configurations, maintaining its predictive accuracy across various scenarios.

Information Integration. In this experiment, we employ addition to combine the state and action information. This approach is intended to mitigate the limitations observed in models that do not consider the full context of state information. We also maintain the original approach for next action prediction, because the next action prediction should depend only on the current state.

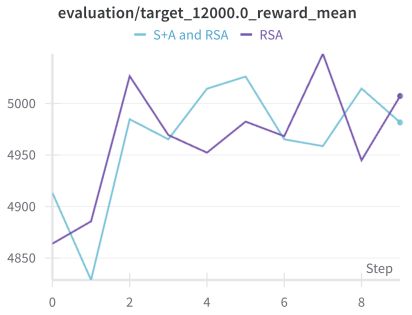

Figure 11: The evaluation rewards of Decision Transformer with/without state information (baseline: without state information) trained on medium-level offline datasets in the Half Cheetah environment.

Figure 11 shows the evaluation rewards of Decision Transformer with/without state information in predicting the next state trained on medium-level offline datasets in the Half Cheetah environment. The results reveal that although the mean rewards are comparable across both settings, incorporating the current state and action information to predict the reward-to-go and next state results in a lower standard deviation after convergence. This suggests that combining input data contributes to learning a more stable policy.

5.4 Experiment 4: V4 Dataset

As described in § 4.5, we train a RL policy from scratch using the actor-critic(AC) (Konda and Tsitsiklis, 1999) and soft actor-critic (SAC) (Haarnoja et al., 2018) approach to obtain a "v4" offline datasets.

However, training reinforcement learning policies in complex environments like MuJoCo can present several challenges, particularly with respect to convergence and reward stability. Unfortunately, despite experimenting with multiple hyper-parameter settings, such as learning rates, discount factors, and the neural network structure, we did not successfully train a stable policy that performs at a medium or expert level on the target MuJoCo tasks.

Therefore, we did not obtain high-quality offline datasets of medium and expert levels. Nonetheless, we have implemented the dataset collection code, which can produce offline trajectories in the same format as the "v2" datasets. Hence, we construct V4 dataset only for the random level.

6 Future Work

6.1 Sequence Modeling for RL

With the rapid development of large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023), which are pre-trained on large corpora and achieve state-of-the-art performance on various natural language generation tasks. Following this line of research, leveraging the generation power of LLMs for sequence modeling on RL tasks might be promising. Alternatively, the pre-training and fine-tuning paradigm could also be adopted in the state, action, and reward sequence modeling.

6.2 Training RL Policies in MuJoCo

Although we have successfully run the SAC model, with training loss decreasing gradually, the gain of evaluation rewards is not stable. The trained policy is not as good as the medium-level of "v2" datasets. Nonetheless, we believe the results will be better via careful hyperparameter selection.

Once we have the "v4" datasets of random, medium, and expert levels. We can compare the performance of Decision Transformer on the MuJoCo-V4 environment after training on either "v2" or "v4" datasets.

6.3 Transfer Learning

In addition, we can pre-train the Decision Transformer on the offline datasets of the chosen three environments and fine-tune the model on another environment. The environment shift can be "v2" to "v4", "medium" level to "expert" level, or even a new MuJoCo task. For transferring to a complete new task, the mismatch of observation/state dimension is of concern. We may solve this problem via network design.

6.4 Online Decision Transformer

Although Decision Transformer is designed for offline reinforcement learning, it is also possible to be used for online RL training. For example, we can use the reward as "loss function" to predict the action. Specifically, we could obtain the softmax-ed reward distribution of each action by performing `env.step` as the ground-truth distribution of the action distribution predicted by the policy. Another possible approach is similar to the idea of DPO (Rafailov et al., 2023).

7 Conclusion

In this work, we explored sequence modeling for reinforcement learning by comprehensively reviewing the related literature and conducting in-depth experiments on the architecture design and tokenization approach of the Decision Transformer and its variants. We utilized the offline datasets of random, medium, and expert levels collected from the MuJoCo-V2 environment and successfully re-implemented, trained, and evaluated the Decision Transformer on the Half Cheetah, Hopper, and Walker2D tasks. As the experimental results demonstrate, we investigated the architecture settings, including hidden state dimensions, number of layers, number of attention heads, and training batch sizes, and found a configuration to balance the trade-off between training efficiency and resource utilization effectively. In addition, the findings of our tokenization experiments verified our hypothesis that the Decision Transformer is robust to variations in trajectory sequences and that providing extra state information for state updates can slightly improve the stability of the model. Moreover, through the exploration of constructing the MuJoCo-V4 offline trajectories, we gained valuable experience in training an RL policy using the soft Actor-Critic method, which deepened the understanding of the lecture materials we learned in the CPSC 533V course. At last, our further analysis and discussions of future work shed light on the directions of sequence modeling for offline reinforcement learning in the current research context.

Task Division

We **contribute equally** and work closely together on research idea discussion, proposal and paper writing, data processing, method implementation, experiments conducting, results analysis, and more. The main focus of each member is shown in Table 3. We all learned a lot in completing this project.

Task	J. Cao	Y. Yin	X. Zhang
Work 1 Code Base		✔	
Work 2 Architecture	✔	✔	✔
Work 3 Tokenization	✔		
Work 4 V4 Dataset		✔	✔
Run Experiments	✔	✔	✔
Results Analysis	✔		✔
Paper Writing	✔	✔	✔

Table 3: Task division of group members.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, abs/1412.3555.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Tom Erez, Yuval Tassa, and Emanuel Todorov. 2011. Infinite-horizon model predictive control for periodic tasks with contacts. In *Robotics: Science and Systems VII, University of Southern California, Los Angeles, CA, USA, June 27-30, 2011*.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3207–3214. AAAI Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

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

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1273–1286.

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

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

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. 2018. Temporal difference models: Model-free deep RL for model-based control. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI blog*.

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

Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. 2024. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, abs/2305.18290.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China. PMLR.

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8:279–292.

Pawel Wawrzynski. 2007. Learning to control a 6-degree-of-freedom walking robot. In *EUROCON 2007-The International Conference on" Computer as a Tool"*, pages 698–705. IEEE.

Paweł Wawrzyński. 2009. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pages 380–390. Springer.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. COMBO: conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28954–28967.

Xiangyuan Zhang, Weichao Mao, Haoran Qiu, and Tamer Başar. 2024. Decision transformer as a foundation model for partially observable continuous control.