# Can Transformers Solve Real-World Reinforcement Learning Problems?

Mridul Khurana *
Virginia Tech
Blacksburg, VA
mridul@vt.edu

Harish Babu Manogaran *
Virginia Tech
Blacksburg, VA
harishbabu@vt.edu

## Abstract

*The approach of treating reinforcement learning as a sequence modeling problem has recently gained traction in the research community. The Decision Transformer [4] algorithm is one such approach that predicts the next optimal action given the history of states, actions, and expected return. In this work, we investigate the practical applicability of decision transformers on tasks that closely reflect real-world conditions. Specifically, we analyze the performance of decision transformers on offline-RL datasets for various tasks provided by the NeoRL framework, which are collected using a conservative policy to avoid exploring risky states that can damage the agent. Additionally, we also evaluate the performance of decision transformers on the adroit robotic hand control tasks provided by D4RL.*

## 1. Introduction

Reinforcement learning (RL) is a powerful technique used to train an agent to perform a sequence of actions in an environment to achieve a specific goal where the agent interacts with the environment by taking actions. It has found various applications in domains of robotics, playing games like Go and Atari, and recommendation systems

Recently, a new paradigm for RL has emerged known as offline reinforcement learning (offline-RL), where the agent learns from a dataset of previously collected experiences, rather than interacting with the environment in real-time. Unlike online RL, where the agent learns from trial-and-error, offline RL is trained on a fixed dataset of transitions, which enables more efficient use of the available data. Offline RL is particularly useful in scenarios where the agent's exploration in the environment is expensive or impossible, making it difficult to collect sufficient experience.

While Reinforcement learning has traditionally been formulated in a Markov decision process (MDP) setting where the agent's state is assumed to fully capture all relevant information about the environment and the agent's policy is optimized using value-based and policy-based RL methods. They are general based on Temporal Difference (TD) [15] learning. Recent works such as Trajectory Transformers [8] and Decision Transformers (DT) [4] have explored the idea of treating reinforcement learning as a sequence modeling problem. Decision transformers take a unique approach and instead of maximizing the total reward achieved, they aim to choose the appropriate action given the cumulative reward to be achieved along with the history of state and action. This approach allows the model to attribute rewards achieved in the current step to critical state-action transitions that occurred several time steps in the past. The advantage of using a transformer-based approach is that it allows the model to selectively focus on relevant parts of the input sequence when generating the output, resulting in improved performance.

Decision transformers have been shown to perform really well in the common RL benchmark tasks such as Atari games [2] and locomotion tasks such as HalfCheetah, Hopper, and Walker from OpenAI Gym [3], its performance in real-world applications has been less studied. This motivates us to try Decision Transformer in near real-world tasks.

In our work, we assess the efficacy of decision transformers in real-world applications on offline-RL datasets. Firstly, we investigate the performance of decision transformers on locomotion tasks like HalfCheetah, Hopper, and Walker available in NeoRL [12]. These datasets are generated by following a conservative policy that avoids risky states to protect the agent from harm and ensure system safety as reflected in the real world. Secondly, we evaluate decision transformers' performance in tasks similar to real-world scenarios such as FinRL [10], CityLearn [17], and Industrial Benchmark (IB) [7] available on NeoRL [12]. We show that Decision Transformers either exceed or match the model-free state-of-the-art offline RL algorithms [1, 9].

*Equal contribution.
Our code is available at https://github.com/mridulk97/DT_real-world

## 2. Background

### 2.1. Transformers

Transformer models are a neural network architecture that uses self-attention mechanisms to selectively weight the importance of input tokens when generating output tokens. These models were introduced by Vaswani et al. [16] and have achieved state-of-the-art performance in various sequence generation tasks, such as language translation, text summarization, and image captioning.

Given an input sequence $\mathbf{x} = (x_1, x_2, ..., x_n)$ and output sequence $\mathbf{y} = (y_1, y_2, ..., y_m)$, the transformer model generates $\mathbf{y}$ by computing a probability distribution over all possible output sequences conditioned on the input sequence, i.e., $p(\mathbf{y}|\mathbf{x})$. This is achieved by computing the product of conditional probabilities of each output token given the previous tokens and the input, i.e., $p(y_i|\mathbf{x}, y_1, y_2, ..., y_{i-1})$.

For our work, we are using GPT-2 architecture [14] which introduces a causal self-attention mask over the original transformer architecture. This modification allows the tokens to be generated in an autoregressive manner.

### 2.2. Offline Reinforcement Learning

Offline reinforcement learning (offline-RL) is the process of training a machine learning model using a dataset of pre-collected experience, without the requirement of any additional interaction with the environment. These approaches have become increasingly popular in recent years, as they allow for more efficient use of data and can be used in scenarios where online interaction with the environment may be costly or impractical.

However, offline RL also poses several challenges, such as the potential for overfitting the training data. Since there is no interaction with the environment we need to ensure that the data collected for training represents the true distribution of the environment as the model training depends on this. Several techniques have been proposed to address these challenges, such as the use of regularization methods, importance weighting, and data augmentation. Overall, offline RL has the potential to enable more efficient and practical use of RL in a variety of applications, but we must carefully consider the design and implementation of offline learning algorithms in order to ensure their effectiveness and reliability.

## 3. Method

In this section, we provide an overview of the Decision Transformers (DT) [4] architecture. Decision Transformer is a type of transformer model that takes a trajectory consisting of past states, actions, and rewards as a condition. In traditional models, the goal is to predict discounted future rewards, but this becomes challenging in the context of the decision transformer. Instead, the model relies on the returns-to-go, represented by $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ which captures the cumulative discounted reward achievable by the agent by following the trajectory from the current time step to the end of the trajectory. This approach assists the model in predicting actions based on the desired future rewards in an autoregressive manner.

We feed the Decision Transformer model with $K$ timesteps, which totals $3K$ tokens containing all the modalities i.e. state, action, and return-to-go. trajectory fed into a decision transformer can be given as below,

$$\tau = \left(\widehat{R}_1, s_1, a_1, \widehat{R}_2, s_2, a_2, \ldots, \widehat{R}_T, s_T, a_T\right) \quad (1)$$

For each modality, the model learns a linear layer in order to obtain the token embeddings. The model uses a convolutional layer for environments that contain visual inputs. In addition to the positional embeddings which are utilized by transformers, Decision Transformer learns an embedding at each time step which is added to the respective token embedding. In our implementation of Decision Transformers, we use GPT-2 [14] model to process these tokens as an autoregressive model and predict the next action.

For training, we adopt a similar approach to the original implementation and sample batches from the dataset each with a sequence of length $K$. The model predicts the action $a_t$ given an input sequence $s_t$ and the loss is averaged at each timestep. In our experiments, we use the mean-squared error across the action spaces for continuous control tasks and cross-entropy loss across discrete action spaces.

## 4. Experimental Results

In this section, we evaluate Decision Transformers for near-real-world tasks on offline RL datasets and compare them to other model-free offline RL algorithms and generally based on TD learning. We provide a comparison to CQL [9] which is state-of-the-art in model-free offline RL. The algorithms we compare are Behaviour Cloning (BC), BCQ [6], CRR [18]. We also compare with one model-based approach BREMEN [11]. The baseline numbers reported are from the original papers and the results are normalized to 100 which represents the expert policy as shown by Fu et al. [5]. The results reported are the normalized mean and standard deviation across 3 runs with different seeds, following the same approach as baselines.

### 4.1. OpenAI Gym

In this section, we reproduce results on various continuous control tasks on OpenAI Gym like Walker2d, Hopper, and HalfCheetah. We are using the D4RL [5] environments for evaluating these tasks similar to how it was done in the original implementation. Table 1 compares our results with

the results reported in the paper. The motivation to reproduce results is to verify the consistency of our implementation of Decision Transformers works well and we can also note that using GPT-2 model [14] as compared to the GPT model [13] used in the original paper has slight improvements of the normalized returns for HalfCheetah and bigger gains on Hopper. Figure 1 show returns evaluated with each update on training on different tasks. We can see that model converges fairly quickly and requires only half the updates as mentioned in the original paper.

| Task | DT (ours) | DT (original paper) |
|---|---|---|
| HalfCheetah | $\mathbf{75.25 \pm 1.33}$ | $74.0 \pm 1.4$ |
| Hopper | $\mathbf{73.5 \pm 0.57}$ | $67.6 \pm 1.0$ |
| Walker2d | $42.17 \pm 0.15$ | $\mathbf{42.6 \pm 0.1}$ |

Table 1. Results for D4RL dataset, reproducing Decision Transformer results in the original paper with ours using GPT-2[14]

## 4.2. NeoRL

NeoRL [12] introduces a new benchmark for tasks to mimic near real-world data and collects policies in conservative and controlled sizes. This differs from other offline RL benchmarks which collects data using highly exploratory policies which may not be the best suited in a real-world scenario where we wish to ensure system safety. It is based on OpenAI Gym APIs, which makes it easy to use. We evaluate the Decision Transformer on different locomotion tasks like HalfCheetah, Walker2d and Hopper. and compare them to other TD [**?**] learning-based algorithms as reported in the original paper. NeoRL also provides various other offline RL datasets like Industrial Benchmark (IB) [7], FinRL [10] and CityLearn [17] that help us evaluate the model in a more real-world setting. Table 2 gives information about the action and state spaces for each of the NeoRL environments. We show that Decision Transformers beat all the baselines for these new near-real-world tasks.

| Environment | Observation Shape | Action Shape |
|---|---|---|
| HalfCheetah-v3 | 18 | 6 |
| Hopper-v3 | 12 | 3 |
| Walker2d-v3 | 18 | 6 |
| IB | 180 | 3 |
| FinRL | 181 | 30 |
| CL | 74 | 14 |

Table 2. Configuration of NeoRL [12] environments. All of the environments have maximum 1000 timesteps except FinRL which has 2516

| Environment | D4RL | NeoRL |
|---|---|---|
| Walker2d | $76.5 \pm 9.5$ | $55.73 \pm 37.05$ |
| Hopper | $73.22 \pm 8.12$ | $58.32 \pm 2.44$ |
| HalfCheetah | $42.38 \pm 3.17$ | $47.59 \pm 23.56$ |

Table 3. Performance of Decision Transformer on datasets provided by D4RL and NeoRL for Mujoco lomotions tasks. Normalized mean and standard deviation for 10 episodes is provided.

| Environment | DT | Baseline |
|---|---|---|
| Walker2d | $75.12 \pm 2.54$ | $76.6 \pm 2.8$ (BCQ) |
| Hopper | $69.63 \pm 2.3$ | $76.6 \pm 1.3$ (CQL) |
| HalfCheetah | $40.89 \pm 3.19$ | $77.4 \pm 1.3$ (CQL) |

Table 4. Comparison of performance of decision transformer on the NeoRL Mujoco locomotion tasks with best-performing baselines in NeoRL [12]. The baseline algorithms are provided in brackets.

### 4.2.1 Mujoco

The NeoRL dataset for Mojuco tasks such as HalfCheetah, Hopper, and Walker is collected using a conservative policy and also an additional dimension is added to the states to provide the information of the position which makes the reward prediction easier for the model since in these environments reward is proportional to the distance moved by the agent.

Table 3 shows the comparison of the performance of DT on the offline datasets provided by D4RL for Mujoco locomotion tasks and the datasets provided by NeoRL for the same set of tasks. As it can be noted, in comparison to the performance of DT on D4RL datasets, the performance on NeoRL datasets where the data is collected by following a conservative policy is not so good, except for HalfCheetah. The limitation set by the conservative policy reduces the diversity of the dataset, which thereby reduces the ability of the DT to learn the environment dynamics. Further, it can be noted the standard deviation of 10 episodes is quite high for NeoRL on Walker2d and HalfCheetah. For each episode, the environment is reset to a new initial state by adding little noise. This shows that DT trained on NeoRL datasets is not so noise prone as it was when it was trained with D4RL which provides much more diverse datasets.

Table 4 provides an overview of how Decision Transformers perform as compared to the baselines. We can see that, Decision Transformers failed to beat the baselines which is mainly due to the dataset being less diverse. This shows that while DT is suitable for real-world environments, the model can still benefit from diverse datasets like any other supervised learning model.

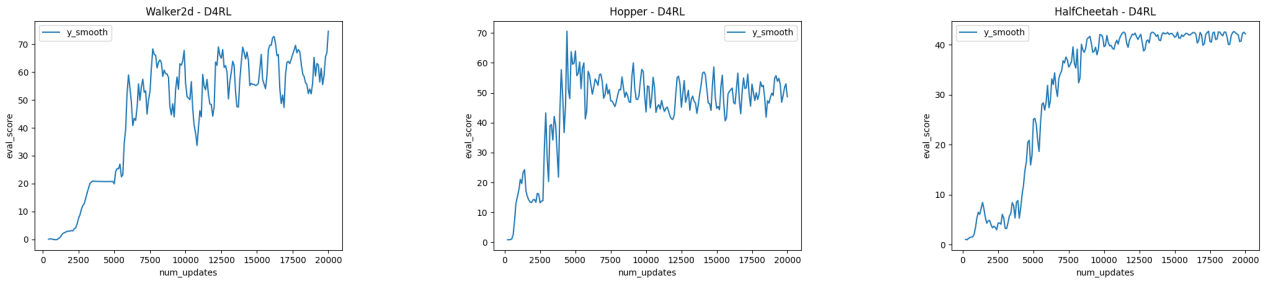


Figure 1. Evaluation score for D4RL Mujoco tasks for a single run with random seed initialization for each update.

| Task | DT | CQL | BC | BCQ | CRR | BREMEN |
|---|---|---|---|---|---|---|
| CityLearn | $\mathbf{120.5 \pm 5.16}$ | $104.7 \pm 5.7$ | $106.7 \pm 1.6$ | $37.5 \pm 11.5$ | $114.2 \pm 2.2^*$ | $110.7 \pm 3.8$ |
| IB | $\mathbf{72.46 \pm 0.7}$ | $15.5 \pm 48.9$ | $9.4 \pm 88.0$ | $-297.9 \pm 80.7$ | $69.7 \pm 0.1^*$ | $-31.5 \pm 113.4$ |
| FinRL | $\mathbf{109.15 \pm 48.47}$ | $57.6 \pm 27.0$ | $48.5 \pm 26.2$ | $16.6 \pm 19.7$ | $43.2 \pm 20.3$ | $70.6 \pm 63.9^*$ |

Table 5. Comparison of performance of decision transformer on the NeoRL. We show that DT beats all the existing baselines as provided in the benchmark. Where * denoted the best result in baselines

#### 4.2.2 CityLearn (CL)

CityLearn is an environment that aggregates the electrical demand curves for different buildings by controlling the energy stored. These energies include Domestic hot water (DHW) and solar power demands that are modeled in the environment. To help monitor and reduce the demand for electricity which has a direct impact on energy costs. In our work, we are using CL collected with the highest policy having 1000 trajectories. Table 5 shows that Decision Transformers beat all the existing baselines. Figure 2 (a) show returns evaluated with each update while training.

#### 4.2.3 Industrial Benchmark (IB)

IB provides a system that simulates characteristics of industrial control tasks like turbines and reactors, with a 6-dimensional vector output for each time step. These six dimensions are gain, setpoint, velocity, consumption, shift, and fatigue. The authors enhance the Markov property by using the last $K = 30$ timesteps as observations. In our work, we are using IB collected with the highest policy having 1000 trajectories. Table 5 shows that Decision Transformers beat all the existing baselines. Figure 2 (b) show returns evaluated with each update while training. We can see that model converges fairly quickly and requires only s few updates.

#### 4.2.4 FinRL

FinRL is a dataset for stock market prediction containing over 30 securities with a history of 10 years of market data. It contains information about the number of stocks owned and other related information about the stocks like price and volume. The goal is to predict the stocks helping in the highest reward. In our work, we are using FinRL collected with the highest policy having 100 trajectories. Table 5 shows that Decision Transformers beat all the existing baselines. Figure 2 (c) show returns evaluated with each update while training.

### 4.3. Androit

We are using D4RL [5] for using the Androit environment. Adroit is a robotic hand control task where the goal of the task is to control a simulated robotic hand to grasp and move objects in a 3D environment. The task is considered challenging due to the high dimensionality of the observation and action spaces, and the need for precise and dexterous hand movements which get one step closer to the real world. Androit has various sub-tasks like pen, hammer, door, and relocate. Table 6 shows the results using Decision Transformer compared to CQL [9]. We can see that Decision Transformer beats CQL in all of the tasks

| Environment | DT | CQL |
|---|---|---|
| Pen-expert-v3 | 112.24 | 107 |
| Hammer-expert-v3 | 127.34 | 86.7 |
| Door-expert-v3 | 104.07 | 101.5 |
| Relocate-expert-v3 | 101.31 | 95 |

Table 6. Comparison of performance of decision transformer on the D4RL adroit tasks with best-performing baseline Conservative Q-learning (CQL) as in the benchmark [5]. Only normalized mean for 3 seeds is provided. Standard deviation is excluded to maintain consistency with baseline results

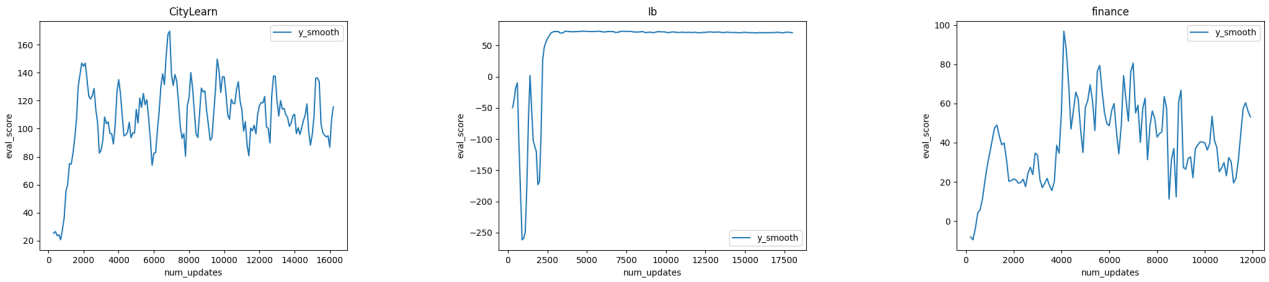

Figure 2. Evaluation score for NeoRL tasks for a single run with random seed initialization for each update.

## 5. Contributions

In the initial work of setting up the codebase for training the model both of us (myself and Harish Babu Manogaran) had an equal contribution. My specific contributions include setting up and automating the pipeline for downloading D4RL Mujoco Datasets as well as NeoRL datasets - FinRL, CityLearn, and Industry Benchmark. Integrating the original implementation of DT model with the updated GPT-2 architecture with our own training pipeline. Running experiments with 3 different initial seeds for D4RL Mujoco tasks - HalfCheetah, Hopper and Walker2d corresponding to Table 1, Table 3 and Figure 1. Also ran experiments for NeoRL tasks - CityLearn, FinRL and IB corresponding to Table 5 and Figure 2. I'm also responsible for maintaining the GitHub repository.

## 6. Conculsion and Future Work

We successfully evaluated Decision Transformer (DT) [4] on various tasks like Industry Benchmark, FinRL, CityLearn, Adroit, and Mujoco which are designed to mimic real-world scenarios. We can see that, Decision Transformers generally match the performance or perform better than the state-of-the-art offline RL models. By doing this, we are addressing the concerns of the original authors whether they perform well on real-world tasks or not. We further see an improvement in performance by changing the backbone transformer model which results in score gains. Therefore the result of our work can be considered as a preliminary analysis to prove the practical applicability of decision transformers.

Decision Transformers have one of the major dependencies on the $return - to - go$ conditioning. They have clearly not explained in the paper how have they chosen the $return - to - go$ condition for each task and why one performs better. In the future, we wish to investigate further with this conditioning and reduce the variance. We also wish to explore the possibility to generalize the algorithm for various continuous control tasks in Mujoco environments. We are planning to build on this work with the future explorations in an upcoming Neurips workshop "Offline Reinforcement Learning Workshop: Offline RL as a "Launchpad""

## References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

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

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

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

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

[7] Daniel Hein, Stefan Depeweg, Michel Tokic, Steffen Udluft, Alexander Hentschel, Thomas A Runkler, and Volkmar Sterzing. A benchmark environment motivated by industrial control problems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[8] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

[9] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[10] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for auto-

mated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2020.

[11] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[12] Rong-Jun Qin, Xingyuan Zhang, Songyi Gao, Xiong-Hui Chen, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:24753–24765, 2022.

[13] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[17] José R. Vázquez-Canteli, Jérôme Kämpf, Gregor Henze, and Zoltan Nagy. Citylearn v1.0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, page 356–357, New York, NY, USA, 2019. Association for Computing Machinery.

[18] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.