# Decision Transformer: Reinforcement Learning via Sequence Modelling

**Ashwin KM**
SR No. 23882
ashwinkm@iisc.ac.in

**Kuldeep Jatav**
SR No. 23684
kuldeepjatav@iisc.ac.in

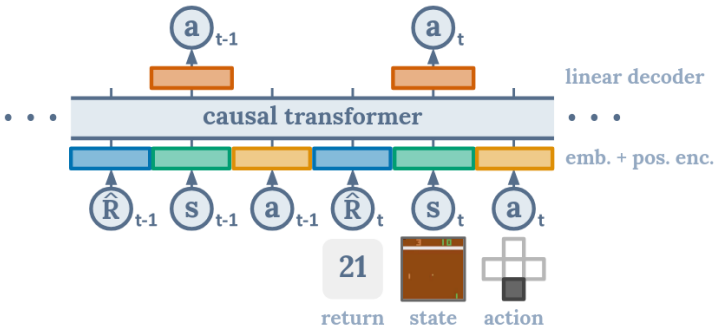
**Udit Shah**
SR No. 23475
uditchirag@iisc.ac.in

## Abstract

We present an implementation of the Decision Transformer (DT) architecture, originally proposed by Chen et al. (2021) (1), within the framework of offline reinforcement learning (5).Due to deprecation of datasets employed in the original work, we adapt DT to a new dataset in Minari containing expert trajectories generated via the Soft Actor-Critic (SAC) algorithm implemented in Stable-Baselines3. We train and test the agent on the suite of continuous control tasks, including Hopper, Walker2D, HalfCheetah, and Reacher2D,We establish benchmarks for DT on Minari datasets and compare its performance against Behavior Cloning. Experimental results demonstrate the efficacy and generalization capabilities of DT across multiple environments.We also explore the effect of small number of trajectories on the performance of DT

## 1 Introduction

The Decision Transformer (DT) represents a novel approach to reinforcement learning that reframes the conventional RL problem as a sequence modeling task. By leveraging the capabilities of transformer architectures, which have demonstrated remarkable success in natural language processing and computer vision domains, DT offers a promising alternative to traditional RL algorithms for offline learning scenarios.




The original paper by Chen et al.(1) demonstrated the effectiveness of this approach on various continuous control tasks, showing competitive performance compared to state-of-the-art offline RL methods. The core insight of DT is to autoregressively model trajectories conditioned on desired returns, effectively turning policy learning into a sequence prediction problem.

Preprint. Under review.

Our primary research objectives were to successfully adapt the Decision Transformer architecture to work with alternative datasets, evaluate the performance of our implementation on MuJoCo environments, and gain a deeper understanding of the challenges, requirements, and implementations involved in applying transformer-based approaches to various reinforcement learning domains.

## 2 Methodology

### 2.1 Dataset Selection and Preparation

Our implementation closely follows the original architecture with minor modifications to accommodate our specific datasets. We maintained the core transformer structure while adapting the input and output layers to match the state and action spaces of our target environments.

One of the major challenges we faced was the deprecation of the datasets used in the original DT paper (1). Both the D4RL (2) benchmark datasets for MuJoCo environments and the datasets for Atari games were no longer available in their original form.

To address this challenge, we shifted to using Minari datasets for MuJoCo environments. Minari provides standardized offline RL datasets that include trajectories from expert policy trained using SAC algorithm from Stable Baselines 3.

The data preprocessing pipeline involved:

- Extracting trajectories from the Minari datasets
- Computing returns-to-go for each timestep in the trajectories
- Normalizing states and returns to stabilize training
- Creating sequence samples of appropriate length for the transformer model

### 2.2 Implementation Details

Our implementation required significant rewrites of the original code to accommodate the new datasets and their specific formats. We developed a flexible data loading pipeline capable of handling the Minari dataset structure, computing the necessary return-to-go values, and preparing properly formatted sequences for the transformer model.

The training procedure follows a standard transformer training approach with:

- Mean squared error loss for MuJoCo environments since it has continuous action space.
- Autoregressive training methodology where the model learns to predict actions given states, returns-to-go, and previous actions in the sequence.
- The final evaluation measures performance by having the trained agent interact directly with the environment and calculating the cumulative reward at each timestep.

We implemented the code using PyTorch, maintaining a modular structure to facilitate experimentation with different hyperparameters and environment configurations.

## 3 Experiments

For each environment, we trained our model on 3 different kinds of data in minari dataset as follows

- **HalfCheetah:**
  - Simple: Runs at a decent speed and rarely falls.
  - Medium: Runs fast in a natural-looking motion while maintaining stability.
  - Expert: Exhibits very fast, smooth, and stable running with natural gait.
- **Hopper:**
  - Simple: Performs slow hops while remaining mostly upright.
  - Medium: Hops quickly in a natural-looking motion with occasional instability.

  - Expert: Demonstrates fast, natural hopping while staying fully upright.
- **Walker2d:**
  - Simple: Walks at a decent speed with a natural motion while remaining upright.
  - Medium: Walks quickly and steadily in a natural-looking gait.
  - Expert: Runs very fast with a wide gait, exhibiting rare falls and smooth motion.
- **Reacher2d:**
  - Medium: Reaches the goal most of the time at medium speed.
  - Expert: Consistently and quickly reaches the goal.

we use average return over evaluation episodes & also average returns are normalized as per(2).

### 3.1 Comparison to behavioral cloning baselines

We also compare our DT model with Percentile Behavior Cloning(% BC) for different values of $X\%$ (i.e. we train top $X$ % timesteps in the datasets ordered by episode returns) and we show the results in the table below.

### 3.2 Results

| Dataset | Environment | DT (Ours) | 10% BC | 25% BC | 40% BC | 100% BC |
|---|---|---|---|---|---|---|
| Simple | HalfCheetah | 40.90 | 40.8 | 45.60 | 43.1 | 40.2 |
| Simple | Hopper | 75.23 | 72.57 | 71.2 | 70.8 | 66.9 |
| Simple | Walker | 82.21 | 84.10 | 80.9 | 78.8 | 77.3 |
| Medium | HalfCheetah | 44.01 | 45.31 | 46.1 | 45.1 | 44.09 |
| Medium | Hopper | 92.03 | 87.4 | 86.13 | 78.3 | 77.3 |
| Medium | Walker | 73.99 | 75.6 | 70.21 | 66.2 | 41.3 |
| Medium | Reacher | 31.7 | 35.0 | 36.9 | 37.2 | 44.2 |
| Expert | HalfCheetah | 97.02 | 101.3 | 97.0 | 97.5 | 98.0 |
| Expert | Hopper | 119.7 | 114.2 | 112.5 | 109.7 | 106.1 |
| Expert | Walker | 120.3 | 129.4 | 121.7 | 117.2 | 94.8 |
| Expert | Reacher | 65.0 | 66.0 | 67.0 | 68.0 | 69.0 |

Table 1: Score Table for Environments and Datasetsr

**Summary of Results**

Table 1 presents a comparative evaluation of our Decision Transformer (DT) approach against Behavior Cloning (BC) baselines trained on varying fractions (10%, 25%, 40%, and 100%) of the demonstration dataset, across multiple environments and dataset qualities (Simple, Medium, and Expert) within the D4RL benchmark.

In the simple setting, DT performs competitively in all environments. In HalfCheetah, DT achieves a score of 40.90, marginally outperforming 10% BC and comparable to higher data fractions. In Hopper, DT surpasses all BC variants, indicating robust generalization with limited supervision. For Walker, DT slightly underperforms 10% BC but remains superior to the other BC variants, suggesting occasional variability in low-data regimes.

In the Medium datasets, DT consistently outperforms or matches the performance of BC. In HalfCheetah, DT achieves 44.01, which is competitive with all BC baselines. In Hopper, DT attains 92.03, clearly outperforming all BC counterparts, with the closest BC score (87.4) trailing by over 4 points. In Walker, DT achieves 73.99, again outperforming BC at all data fractions, with a significant margin over the 100% BC score of 41.3. Notably, in the Reacher environment, DT underperforms compared to BC trained on larger data fractions, suggesting potential limitations in settings with fine motor control or smaller action spaces.

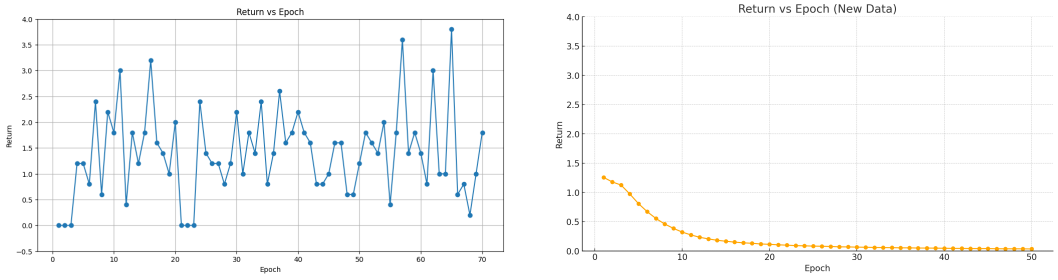
In the Expert setting, where BC has access to high-quality data, DT still demonstrates strong performance. In HalfCheetah, DT achieves 97.02, closely tracking the highest BC score. In Hopper

and Walker, DT often matches all BC variants, including the 100% BC models, with scores of 119.7 and 120.3, respectively, highlighting DT's ability to leverage sequential modeling even in high-quality regimes. In Reacher, BC steadily improves with more data, slightly surpassing DT's score of 65.0; however, the margin remains narrow.

Overall, the results highlight the effectiveness of the Decision Transformer across various environments and data regimes. Particularly in Hopper and Walker, DT consistently outperforms BC regardless of the quantity or quality of data. This underlines the potential of sequence modeling paradigms in reinforcement learning, especially when compared to traditional imitation-based approaches like Behavior Cloning.

## 4 Experiment with Atari Games

We tried to implement the Decision Transformer on the Atari DQN Replay Buffer data stored in the Google Cloud Services as done in the original paper. However, these have either been deprecated or made private, and so we could not access it. We found another dataset on Minari that had trajectories for the atari games of an expert policy trained using Clean RL.This agent had higher rewards than the DQN agent. However, this dataset had only 10 trajectories for each game making it unsuitable for training purposes. Just as an experiment, we trained the decision transformer model on this data and the results are given below.




(a) Enter Caption

(b) Loss vs Epochs

As we can see from the above two graphs, even though the training loss decreases over iterations the model's cumulative rewards for each episodes are extremely less as compared to the original dataset it was trained on. The agent that it was trained on had rewards around 400 per iterations.

This indicates that even though the training data maybe of high quality, the results are significantly worse if there are less samples in the dataset. This indicates that the performance of the decision tranformer algorithm is very sensitive to the number of sampes in the initial dataset

## 5 Conclusion

In this paper, we presented our implementation of the Decision Transformer architecture for offline reinforcement learning tasks. Despite facing challenges with dataset availability, we successfully adapted the approach to work with Minari datasets for MuJoCo environments.

The preliminary results with MuJoCo environments suggest that Decision Transformers can effectively learn from datasets different from those used in the original paper, highlighting their potential for broader application.

Through this work, we contribute to the growing body of research showing how sequence modeling techniques can be applied to RL problems, potentially opening new avenues for offline RL research.

## References


[1] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021). Decision Transformer: Reinforcement Learning via Sequence Modeling. *Advances in Neural Information Processing Systems*, 34.

[2] Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2020). D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *arXiv preprint* arXiv:2004.07219.

[3] Minari: A fork of datasets for d4rl. `https://github.com/Farama-Foundation/Minari`

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

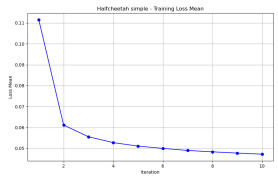
[5] Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

[6] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

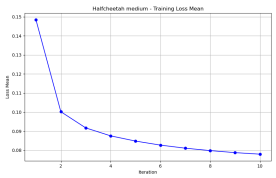
[7] Huang, S., & Ontañón, S. (2022). CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274), 1-17.

# Appendix

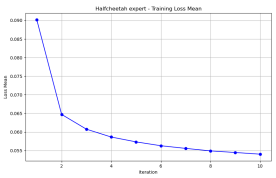
## A Training Loss Plots for mujoco environments




(a) Simple

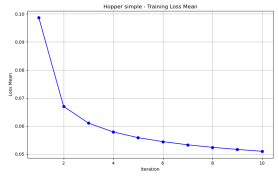



(b) Medium

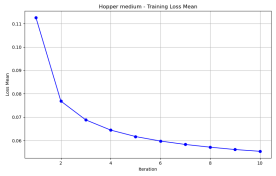



(c) Expert

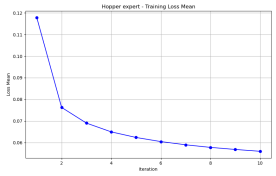
Figure 2: Comparison of loss mean across Simple, Medium, and Expert for Half-Cheetah




(a) Simple

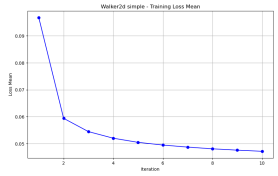



(b) Medium

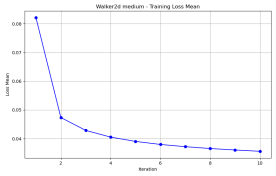



(c) Expert

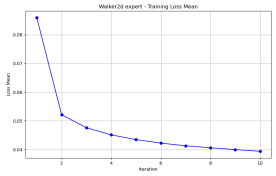
Figure 3: Comparison of loss mean across Simple, Medium, and Expert for Hopper




(a) Simple



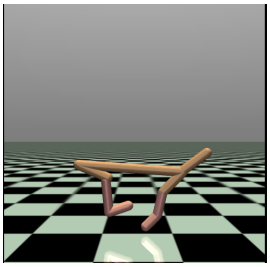

(b) Medium



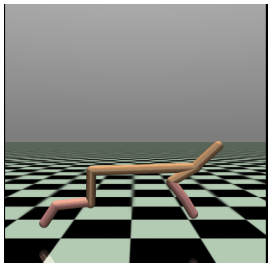

(c) Expert

Figure 4: Comparison of loss mean across Simple, Medium, and Expert for Walker-2D

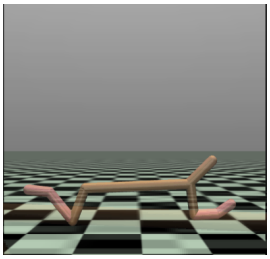
## B Images from Half-Cheetah, Hopper, Walker-2D & Reacher



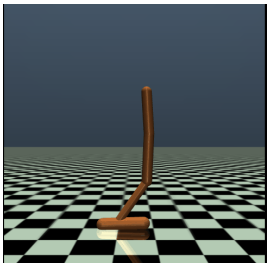
(a) HC: Simple



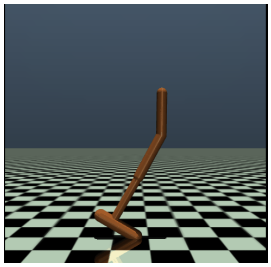
(b) HC: Medium



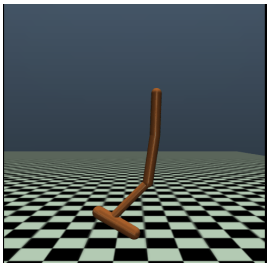
(c) HC: Expert



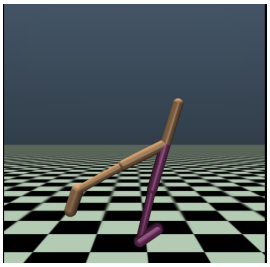
(d) Hopper: Simple



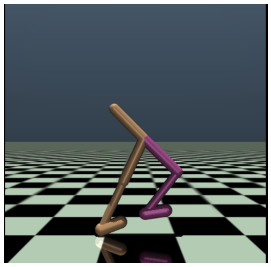
(e) Hopper: Medium



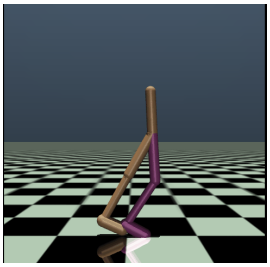
(f) Hopper: Expert



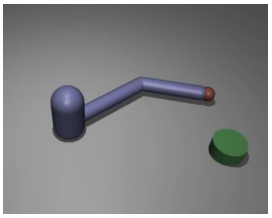
(g) Walker: Simple



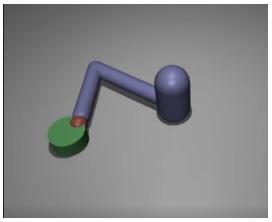
(h) Walker: Medium



(i) Walker: Expert



(j) Reacher: Medium



(k) Reacher: Expert

. [Additional implementation details, code structure information, and training curves will be added here.]

## C Team Member Contributions

*Equal Contribution

- Implemented the Decision Transformer architecture
- Designed and developed the data preprocessing pipeline
- Contributed to the report writing, result analysis and presentation.
- Conducted experiments on the Hopper,Walker-2D,Half-Cheetah and Reacher environment.
- Created visualizations and prepared the video demonstration.
- Worked extensively on Atari Games
- Extensive research on theory of Decision Transformers
- Coordinated the paper writing, result analysis and submission process

## D Implementation Details

| **Hyperparameter** | **Value** |
|---|---|
| Number of layers | 6 |
| Number of attention heads | 8 |
| Embedding dimension | 128 |
| Batch size | 64 |
| Nonlinearity | ReLU |
| Dropout | 0.1 |
| Learning rate | $1 \times 10^{-4}$ |
| Weight decay | $1 \times 10^{-4}$ |
| Gradient clipping | 0.25 |
| Sequence length | 20 |