Exploring the Capabilities of Decision Transformers

Anne Castille Buisson
acb618@mit.edu

Gabriel Afriat
afriatg@mit.edu

Sara Pasquino
pasquino@mit.edu

Abstract

This project investigates the potential of Decision Transformers in reinforcement learning, focusing on their application in gaming scenarios. Initially, we aim to replicate the findings of Chen et al. [1] by applying the Decision Transformer to the Atari game Breakout [2] and the Key-to-Door environment [3]. Subsequently, we conduct a thorough analysis of the model's strengths and limitations by varying important hypeparameters specific to Decision Transformers. Finally, the Decision Transformer is adapted for Pac-Man [2], to test its generalizability across different gaming environments. Success is assessed based on evaluation returns, sample efficiency, and the stability of the learning process, ensuring a comprehensive understanding of the model's capabilities and areas for improvement.

1 Introduction

Decision Transformers (DT) utilize the transformer architecture, traditionally used in natural language processing, to solve reinforcement learning (RL) tasks. Unlike traditional RL methods, DT treat RL as a sequence prediction problem, directly predicting optimal action sequences from historical data. This method leverages transformers' self-attention mechanisms to handle long-term dependencies effectively and bypasses the need for bootstrapping—typically used in long-term credit assignment problems — thus avoiding a common instability in RL known as the "deadly triads" [4].

2 Problem Description

Let K be the context length and f_θ a decision transformer parametrized by θ. Let $f_{\theta_s}^{\text{embed\_s}}$, $f_{\theta_a}^{\text{embed\_a}}$, $f_{\theta_r}^{\text{embed\_r}}$ the embedding function of the states, actions and returns to go (future expected/desired rewards). f_θ uses the K previous embedding of (return to go, state, action) to predict the action:

$$\hat{a}_{t+1} = f_\theta\big(\big(f_{\theta_r}^{\text{embed\_r}}(r_{t-i}), f^{\text{embed\_a}}(a_{t-i}), f_{\theta_s}^{\text{embed\_s}}(s_{t-i})\big)_{i\in[K]}\big) \tag{1}$$

The loss function that Chen et al. [1] tries to minimize is the cross-entropy loss (in the discrete action space setting) between the predicted action and the true one.

Conservative Q-Learning (CQL) [5] is a state-of-the-art offline learning method, and main competitor of DT. The goal of CQL is to minimize:

$$\begin{aligned}\min_\theta\ & \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi(a|s)}[Q_\theta(s,a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\beta(a|s)}[Q_\theta(s,a)] \\ & + \mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\Big(Q_\theta(s,a) - r(s,a) - \gamma\max_{a'} Q_{\theta_{\text{target}}}(s',a')\Big)^2\right]\end{aligned} \tag{2}$$

with π the current policy and π_β the behavior cloning policy.

While the first second term minimizes the temporal difference error and fits the Q-value on the dataset, the first term tries to minimize the Q-value for unseen (state, action) pairs to avoid overestimation.

3 Project Objectives

Our research focuses on the following key objectives:

- **Validation of Results:** Replicate Chen et al.'s experiments on the Atari game Breakout to confirm their reported outcomes.
- **Ablation Study:** Explore how changes in the model's hyperparameters, specifically the size of the dataset and context length, affect the performance of the DT.
- **Stability Analysis:** Assess the consistency and reliability of the model's performance across multiple seeds.
- **Comparative Benchmarking:** Conduct a comparative analysis of DT against Conservative Q-Learning (CQL), a state-of-the-art offline learning algorithm.
- **Generalizability to new games:** Extend the application of DT to a new game, Ms. Pacman, to evaluate the model's adaptability and performance across different gaming scenarios.
- **Long-term credit assignment:** Train an agent on the Key-to-Door environment using PPO to collect data. The data is used to train a DT offline to assess its performance on a long-term credit assignment problem.

4 Experimental Setup

In the original DT paper [1], all the results are obtained using 3 seeds (123, 231 and 321). For more reliable results, we used **5 seeds** (0,1,2,3,4). In the following, we will report the **mean** and **mean absolute deviation** across 5 seeds.

In this project, we used three environments: Breakout, Ms. Pacman and Key-to-Door. The state and action space, goal of the games and rewards are described in Appendix A.1, A.2 and A.3 respectively.

5 Validation of Model Performance

We first attempted to replicate the experiment conducted by Chen et al. using the Decision Transformer on Breakout. Apart from the seeds (use of 5 different seeds instead of 3 for more reliable results), we used the same hyperparameters as in [1]. The evaluation results, as shown in Figure 1, were then normalized based on the protocol by Hafner et al. [6], where a score of 100 represents the performance of a professional gamer and 0 a random policy. We obtained a normalized average return of **208.9 ± 32.5**, slightly below the **267.5 ± 97.5** reported by Chen et al., likely due to variations in training data and model initialization. The results we obtain are within the uncertainty bars provided by [1]. In addition, we also find that DT still outperform the professional gamer benchmark (score above 100), demonstrating its effectiveness.

	Decision transformer (seed 0,1,2,3,4)	Decision transformer (seed 123, 231, 312)
Breakout	208.9±32.5	267.5 ± 97.5

Table 1: Comparison of normalized evaluation return of DT obtained by Chen et al. (seed 123, 231 and 312) and us (seed 0,1,2,3,4)

.

6 Ablation Study of Decision Transformers

For all the ablation studies below, we use for the remaining hyperparameters the default values for Breakout as mentioned in [1].

6.1 Impact of Dataset Size

The first ablation study was conducted to evaluate the impact of varying dataset sizes, i.e. number of (state, action, reward) steps, on the performance of DT. As depicted in blue in Figure 1, training time increases significantly while evaluation return also sees substantial improvement and training loss keeps decreasing. Increasing the size of the dataset helps improving training and generalization but requires a higher computational cost (each epoch requiring more training steps).

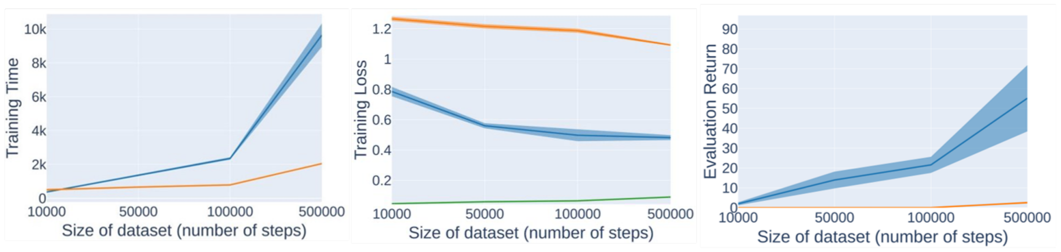

Figure 1: Impact of Dataset Size on DT (blue) and CQL (timing and return in orange, conservative loss in orange and TD loss in green) for the Breakout environment

6.2 Impact of Context Length

The second set of experiments explored the effect of context length K (the number of past returns to go, actions and states considered by the model) on Decision Transformer performance, with results presented in Figure 2. As K increases, the transformer architecture gets bigger, and the training time increases. Training loss consistently decreases with increased context length. However, the evaluation return peaks at a context length of 10, demonstrating high performance at moderate lengths where the model benefits from sufficient historical data without overfitting.

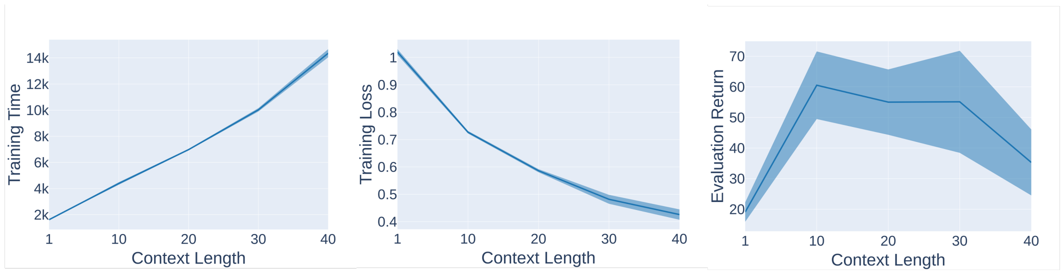

Figure 2: Impact of Context Size on DT for the Breakout environment

7 Stability Analysis of Decision Transformers

While training loss showed minimal variation across different seeds, indicating similar training behaviors, more substantial variations were observed in evaluation returns. The method being offline, the agent can only be trained on the given dataset. For the optimal context length, the agent is able to perform very well when the trajectory is in distribution, but struggles to adapt to those that are out-of-distribution, hence the high variance.

8 Comparative Benchmarking: DT and CQL

	Decision transformer	Conservative Q-Learning
Breakout	60.5±11.1	2.54±0.4
Ms. Pacman	1574.4±162.7	1135.4±341.5

Table 2: Evaluation return for DT and CQL on Breakout and Ms. Pacman

As seen on Table 2, DT beats CQL consistently on the two games we tried (Ms. Pacman and Breakout). The poor perfomance of CQL on Breakout can be explained by the lack of hyperparameter search due to time and resource constraints. In addition, more epochs could also have improved the numbers. When looking at Figure 4, the TD loss seems be going up, which means the Q value probably didn't have enough time to converge, and the conservative loss keeps going down. CQL might have needed more time to be more conservative to the unseen (state,action) pairs while being accurate on the dataset information.

9 Generalizability to new games: Decision Transformers for Ms. Pacman

An important aspect of our research was to determine the generalizability of DT when applied to new games such as Ms. Pacman. To optimize the Decision Transformer for Ms. Pacman, we first needed to find a good context length and target return through **hyperparameter search** (see Appendix A.4).

We then conducted similar ablation studies as in sections 6.2 and 6.1 to validate the model's performance. The results are shown in Figure 3. Training time increases for both models as the dataset size expands, but DT consistently shows returns above those obtained from CQL. With proper fine-tuning of context length to avoid overfitting, DT improves in handling complex decision sequences compared to CQL, as indicated by the higher evaluation return around a context length of 10.

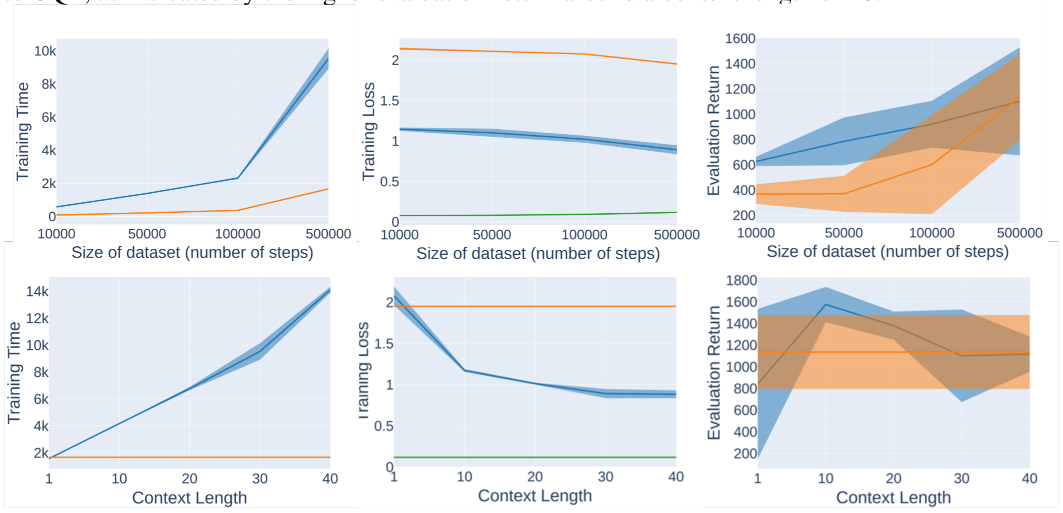

Figure 3: Impact of Dataset Size (top) and Context Length (bottom) on DT (blue) and CQL (timing and return in orange, conservative loss in orange and TD loss in green) for the Ms Pacman environment

10 Long term credit assignment problem

While Ms. Pacman and Breakout are interesting RL tasks to investigate, Chen et al. [1] insist on the capacity of DT to handle long-term credit assignment problems where the reward is sparse and requires a long sequences of actions before being obtained. In order to test this hypothesis, we considered the Key-to-Door [3] environment as described in Appendix A.3. Given the absence of dataset, we first had to create our own by training an expert agent.

10.1 Training an expert on Key-to-Door

While some instances of the Key-to-Door environment are easy (Key-to-Door 5x5 for instance), some are intermediate (Key-to-Door 6x6) and some of the bigger ones are much harder to solve (Key-to-Door 8x8 and Key-to-Door 16x16). Training a PPO agent directly on this environment wouldn't work. Therefore, we proceed as follows:

1. Train a PPO agent on Key-to-Door 5x5 for 100,000 steps
2. Further train the same agent on the next larger instance of Key-to-Door for 100,000 steps

This process (see Appendix A.5) enables us to learn an expert on the Key-to-Door 8x8 environment (16x16 remaining too hard to solve). We then collect data from that expert agent on Key-to-Door 8x8.

10.2 Training the DT on Key-to-Door 8x8

Unfortunately, the DT trained on the newly created dataset wasn't able to solve the Key-to-Door 8x8 environment. We believe that a larger hyperparameter search and/or longer training might have been needed here (but wasn't possible due to time constraints and limited computational resources). In [1], the authors explain using a context length of the size of the trajectories for the Key-to-Door environment. However, our dataset being suboptimal, some trajectories were very long. In addition, due to computational limits, a context length greater than 30 was not tractable (see Appendix A.6)

11 Contributions

- Gabriel Afriat:
  - Setting up the Decision Transformers on Supercloud **(existing code, https://github.com/kzl/decision-transformer/tree/master/atari)**
  - Adapting the DT pipeline to the Key-to-Door environement (new state embedding for the 7x7x3 images instead of 4x84x84, different data pipeline, different evaluation pipeline as the environment is not from Atari) **(new code)**
  - Setting up CQL on Supercloud **(existing code, d3rlpy)**
  - Running PPO on Key-to-Door 5x5, then 6x6 and 8x8 **(original idea)**
  - Data creation and processing from the PPO expert **(new code)**
  - Setting up hyperparameter search on SuperCloud **(new code)**
  - Attempts to train PPO on Key-to-Door 16x16 using a dense reward function **(minigrid, new code for reward function)**
  - Attempts to train DQN on Key-to-Door 16x16 with manually added trajectories **(stable_baselines3, minigrid, new code for replay buffer)**
  - Setting up ablation study on SuperCloud **(new code)**
  - Worked on the slides and report with other teammates
- Anne Castille Buisson:
  - Setting up the Decision Transformers on Colab **(existing code, https://github.com/kzl/decision-transformer/tree/master/atari)**
  - Setting up ablation study on Colab **(new code)**
  - Setting up hyperparameter search on Colab **(new code)**
  - Attempts to train PPO on Key-to-Door 16x16 using a dense reward function **(minigrid, new code for reward function)**
  - Worked on the slides and report with other teammates
- Sara Pasquino:
  - Setting up the Decision Transformers on Supercloud **(existing code, https://github.com/kzl/decision-transformer/tree/master/atari)**
  - Setting up hyperparameter search on Colab **(new code)**
  - Setting up ablation study on Colab **(new code)**
  - Attempts to train PPO on Key-to-Door 16x16 using a dense reward function **(minigrid, new code for reward function)**
  - Worked on the slides and report with other teammates

A Appendices

A.1 Breakout Environment

The Breakout environment - Figure 4 (left) - is characterized by a state space $\mathcal{S} \subset \mathbb{R}^{4\times 84\times 84}$. Each state corresponds to a stack of the four previous comprehensive views of the game, each measuring 84x84 pixels. The action space $\mathcal{A}$ consists of four discrete actions: NOOP (no operation), FIRE (launch the ball), MOVE RIGHT, and MOVE LEFT. The reward system in Breakout gives points for breaking bricks, with the amount of points varying based on the brick's color.

A.2 Ms. Pacman Environment

The Ms. Pacman environment - Figure 4 (middle) - also has a state space $\mathcal{S} \subset \mathbb{R}^{4\times 84\times 84}$. Each state corresponds to the four previous frames of the game, each 84x84 pixels. The action space $\mathcal{A}$ includes nine possible movements: NOOP, UP, DOWN, LEFT, RIGHT, UPLEFT, UPRIGHT, DOWNLEFT, and DOWNRIGHT. In terms of rewards, the player earns points for each pellet collected and for ghosts consumed after Ms. Pacman has eaten a power pellet.

A.3 Key-to-Door Environment

The Key-to-Door environment - Figure 4 (right) - has a state space $\mathcal{S} \subset \mathbb{R}^{7\times7\times3}$. Each state corresponds to the agent's current field of vision, which is an RBG 7x7 image. The action space $\mathcal{A}$ includes seven possible actions: [0] Turn Left, [1] Turn Right, [2] Move Forward, [3] Pick Up an object, [4] Drop object, [5] Activate an object, [6] Done. In terms of rewards, the player earns one point if they use the key to open the door and then get to the goal.

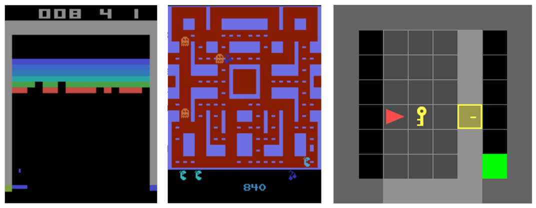

Figure 4: Breakout environment (left), Ms. Pacman environment (middle) and Key-to-Door (right)

A.4 Tuning Target Return and Context Length on Pacman

This involved testing various configurations across one seed (seed = 0). In order to reduce the hyperparameter search:

- We first fix the target return to 2000 (arbitrary choice based on some previous runs). We then tried to tune the context lengths $K \in \{1, 10, 20, 30, 40, 50\}$. We obtain that $K = 30$ is the best choice here.
- We first fix the context length to the previously found value $K = 30$ and we then try to tune the target return $T \in \{1500, 1600, 1700, 1800, 1900, 2000\}$. We obtain the best return for $T = 1600$ here.

During this hyperparameter search, all the other hyperparameters are fixed to the Breakout default values, as described in [1].

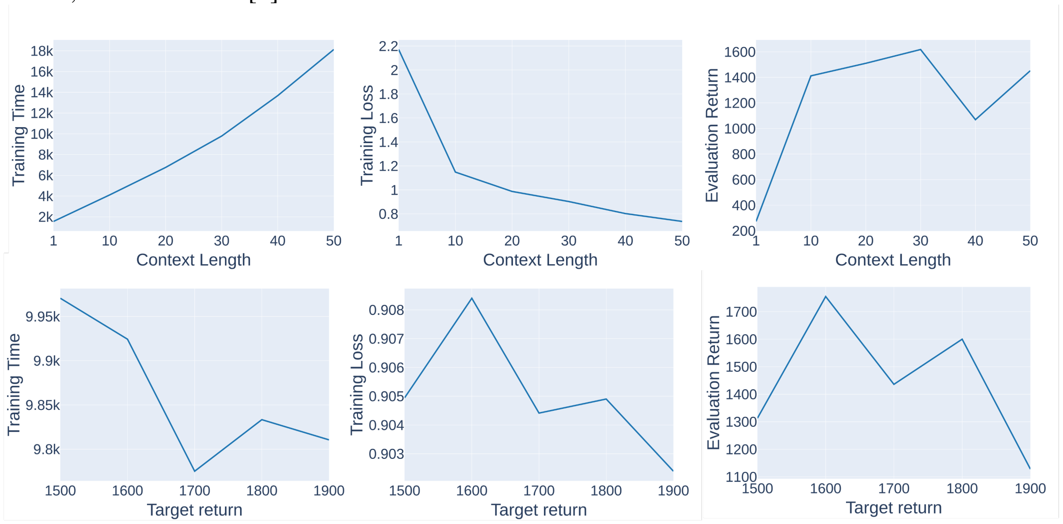

Figure 5: Results from finetuning the Decision Transformer on Pacman, showcasing the determination of optimal context length (top) and target return (bottom).

A.5 PPO on Key-To-Door

The training on PPO used the following hyperparameters:

- Batch size: 256
- Discount factor: 0.99
- Learning rate: 1e-3

- gae-lambda: 0.95
- entropy-coef: 1e-2
- value-loss-coef: 0.5
- max-grad-norm: 0.5

A.6 Decision Transformers training on Key-To-Door

Once the expert agent trained, we collected 300,000 steps using 5 different seeds (0,1,2,3,4) (for a total of 1,500,000 steps).

For each seed, we noticed that a few trajectories had around 10 steps. Because of that, we first tried to train a Decision Transformer on 300,000 steps using K=10. This Decision Transformer was unfortunately unable to learn.

We noticed that for all seeds, most trajectories (more than 250,000 steps) had more than 30 steps. In addition, because of prior experiments, we knew that K=30 was also a limit on our computational resources (K>30 requiring a very slow training). Therefore, we decided to only keep the 250,000 steps for each seed that had more than 30 steps and to use a context length of K=30. The results we obtained are below:

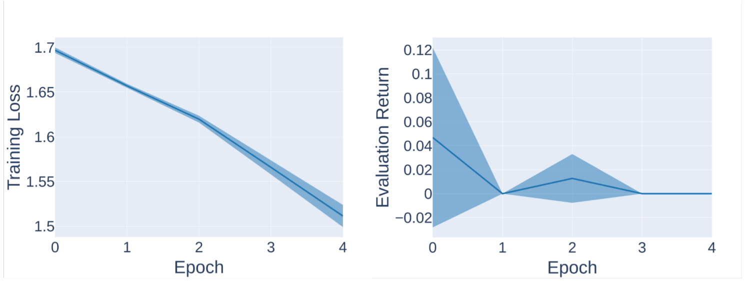

Figure 6: Results of DT on Key-to-Door 8x8

References

[1] 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. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf`.

[2] Atari 2600 vcs rom collection. `https://www.atarimania.com/rom_collection_archive_atari_2600_roms.html`.

[3] Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, Éric Moulines, Marcus Hutter, Lars Buesing, and Rémi Munos. Counterfactual credit assignment in model-free reinforcement learning, 2021.

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

[5] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf`.

[6] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022.