DECISION TRANSFORMER: REINFORCEMENT LEARNING VIA SEQUENCE MODELING

Youssef Fathi,
CS 885: Reinforcement Learning
Winter 2022

Presented to: Prof. Pascal Poupart

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

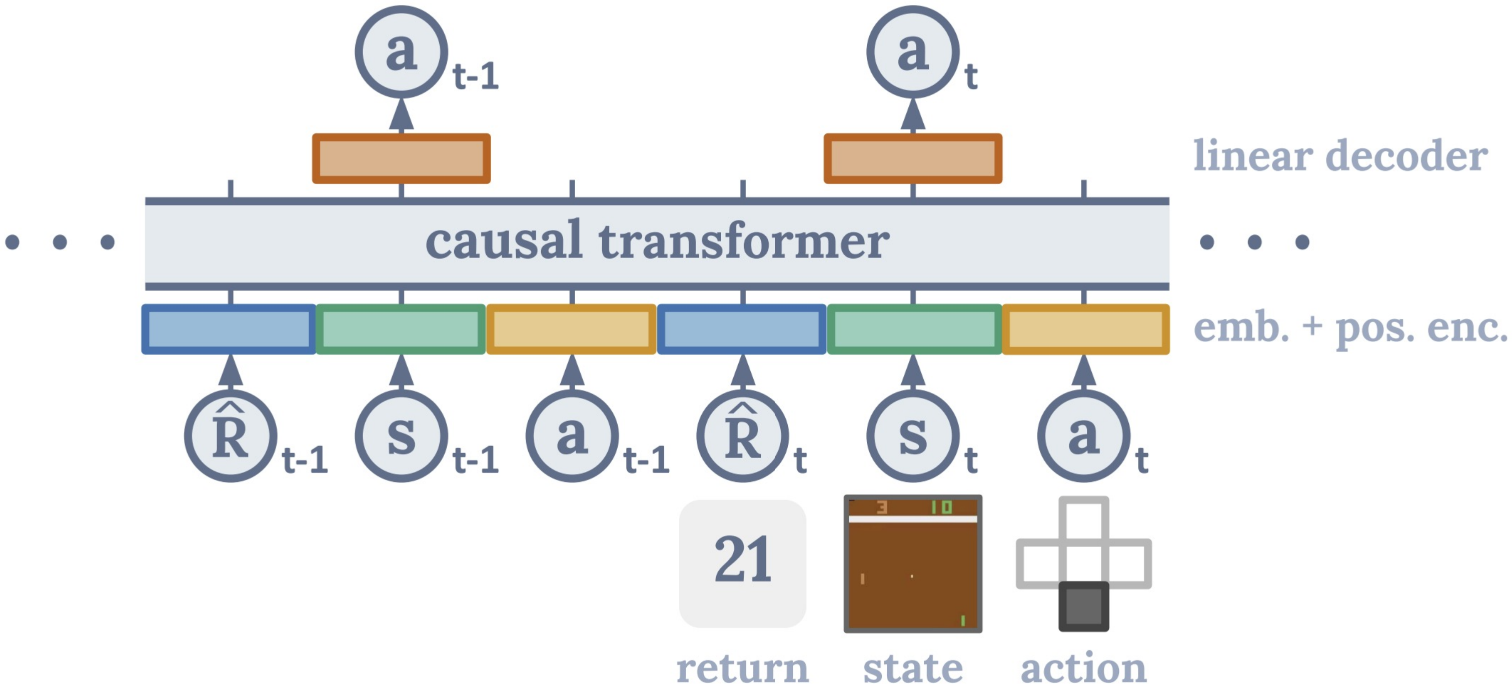
Outline

- Introduction
- Background
- Methodology
- Evaluation
- Discussion
- Conclusion

INTRODUCTION

Introduction

- Supervised RL
- Model-Free
- Offline RL
- Sequence Modelling



UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

BACKGROUND

Background: Markov Decision Process (MDP)

Model-based

States: S.

Actions: A.

Transition Model: $P(s_t|s_{t-1}, a_{t-1})$

Reward Model: $\mathrm{R}(s_t, a_t)$

Discount Factor: $0 \leq \gamma \ \leq 1$

Horizon: h

Model-free

States: S.

Actions: A.

~~**Transition Model:**~~ $\cancel{P(s_t|s_{t-1}, a_{t-1})}$

~~**Reward Model:**~~ $\cancel{\mathrm{R}(s_t, a_t)}$

Discount Factor: $0 \leq \gamma \ \leq 1$

Horizon: h

Background: Online Reinforcement Learning (RL)

Target: Maximize expected sum of discounted rewards.

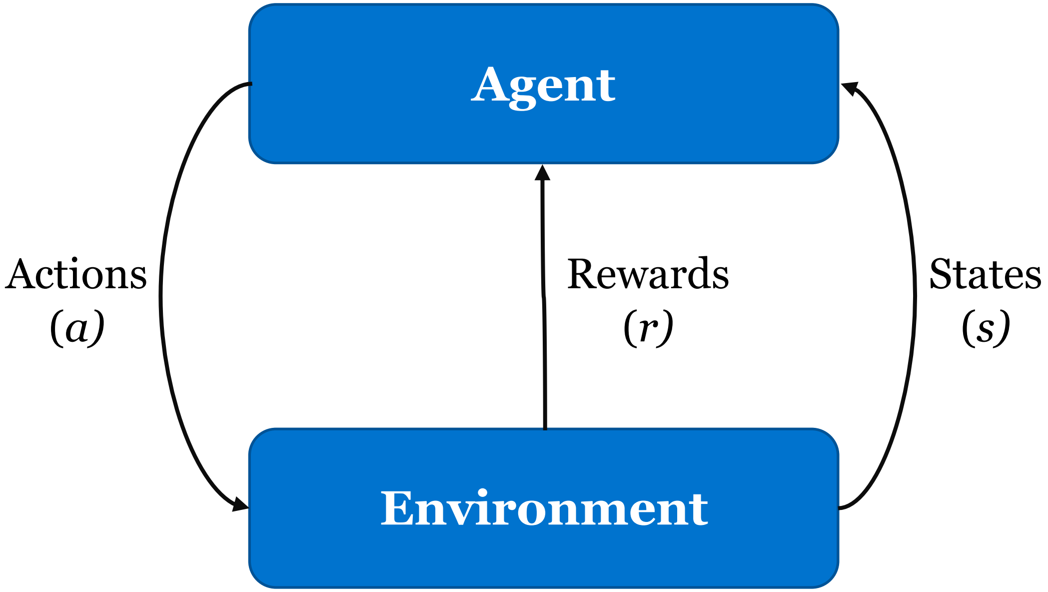
$$E[R] = \sum_i \gamma^i r_i$$

Method: Using temporal difference in bellman backups to estimate optimal value function [1,2].

$$Q_i(s,a) = Q_{i-1}(s,a) + \alpha(r_i + \gamma\, Q_{i-1}(s',\pi(s')) - Q_{i-1}(s,a))$$

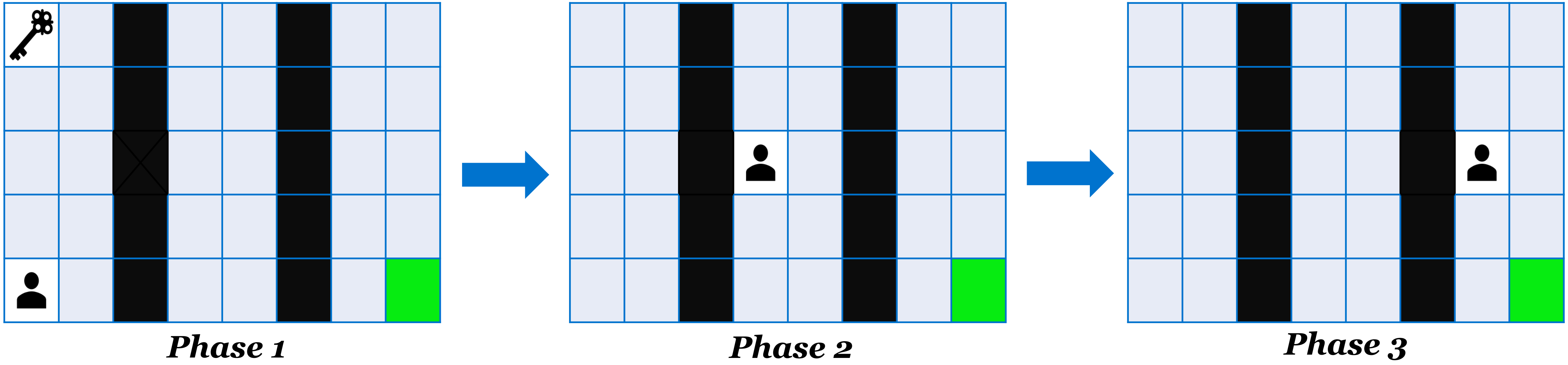
Challenges:

- High cost/risk of interaction with environment
- Credit assignment problem due to discounting in bellman backups.



$$\tau = (s_0, a_0, r_0) \rightarrow (s_1, a_1, r_1) \rightarrow (s_2, a_2, r_2) \rightarrow ..$$

Background: Credit Assignment Problem (Key-to-Door Env. [27])



Problem: Assigning delayed rewards to their originating actions.

Possible Solution: State association [16,17,18,19]

Background: Offline (Batch) Reinforcement Learning

- **Objective**: Learning from a fixed dataset without further interactions with the environment.

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots \dots, s_n, a_n, r_n)$$

Pre-generated

- **Popular Examples**: DDPG [4].

- **Main Challenge:** Distribution Shift (Extrapolation Error) [6].

$$Q_i(s,a) = Q_{i-1}(s,a) + \alpha(r_i + \gamma\, Q_{i-1}(s', \pi(s')) - Q_{i-1}(s,a))$$
$$\pi(s') = argmax_a\, Q(s', a)$$

$\pi(s')$ chooses rarely visited (s', a')

- **Solutions Proposed:**
  - Constrain policy action space [6,7]
  - Incorporate value pessimism [6,8]

Background: Supervised RL

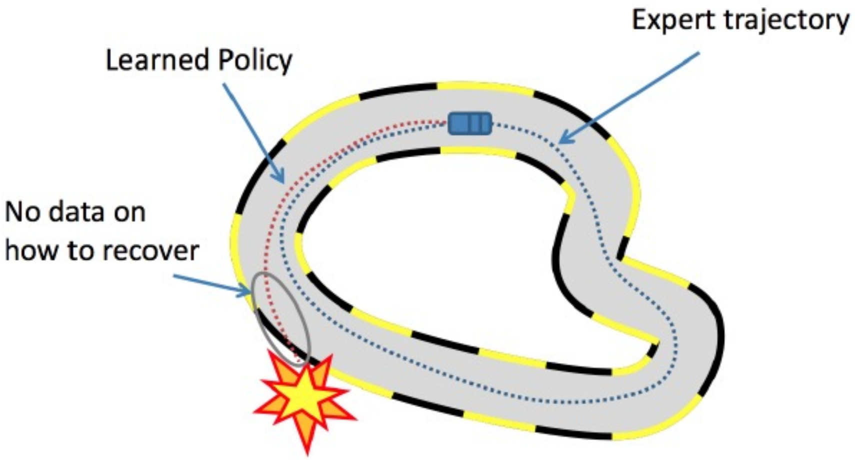
Imitation Learning: imitating the behaviour observed in existing trajectories.

- Behavioural Cloning (Basic Version) : using supervised losses to map existing *states* to *actions* with *no regards to rewards. [11]*

$$f(s) = a, \qquad (s,a) \epsilon \{(s_1,a_1),(s_2,a_1),\dots,(s_n,a_n)\}$$

Drawbacks

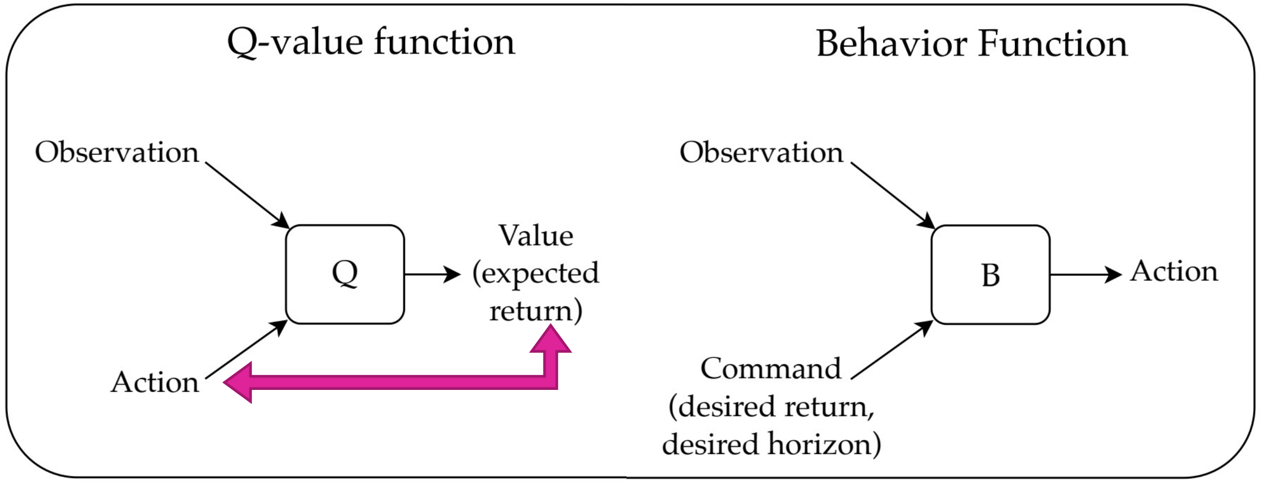
- Impossible to generalize to new scenarios.
- Requires large amount of optimal (expert) actions in the trajectories
- Assumes state-action pairs are i.i.d.



Source: http://web.stanford.edu/class/cs234/slides/lecture7.pdf

Background: Supervised RL

Upside-Down RL [12]: trains agents to follow commands such as "obtain so much total reward in so much time."



Variants

Kumar et.al [13]	Ghosh et al. [14]	Paster et al. [15]
• Fully Offline RL • Reward Conditioning	• State Conditioning	• Online RL • LSTM with State Conditioning

Supervised Objective

$$B = argmin_B \sum_{t_1, t_2 \in \tau} L(B(a_{t_1}, s_{t_1}, d^r, d^h), a_{t_2})$$

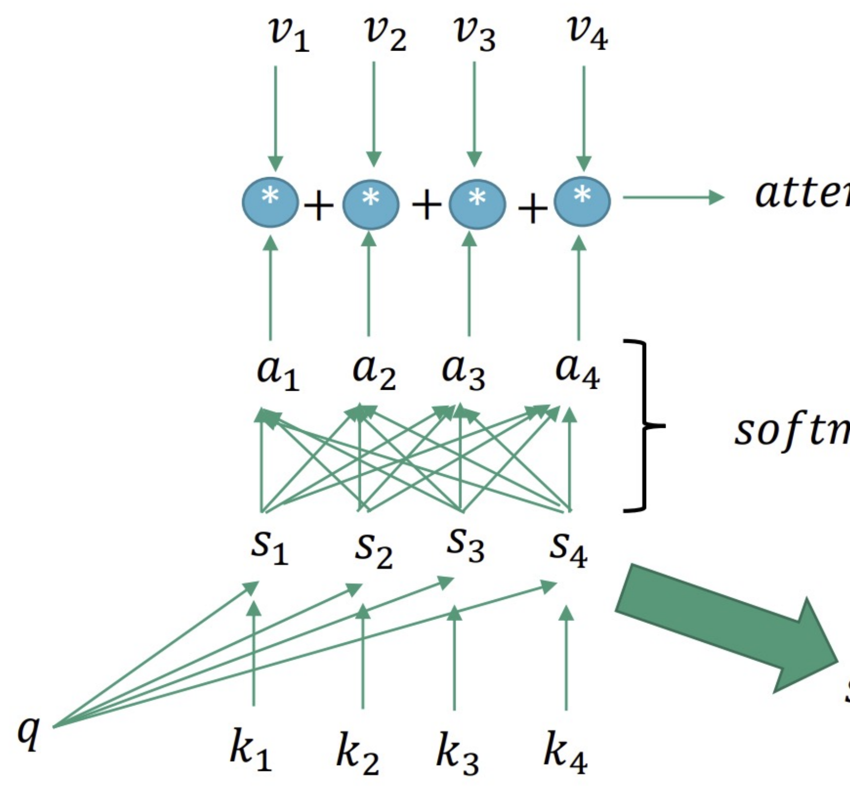
Background: Attention

"The *cat* drank the milk because **it** was hungry."

"The cat drank the *milk* because **it** was sweet."

Credit: https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

Background: Attention



$$attention\ value = \sum_i a_i v_i$$

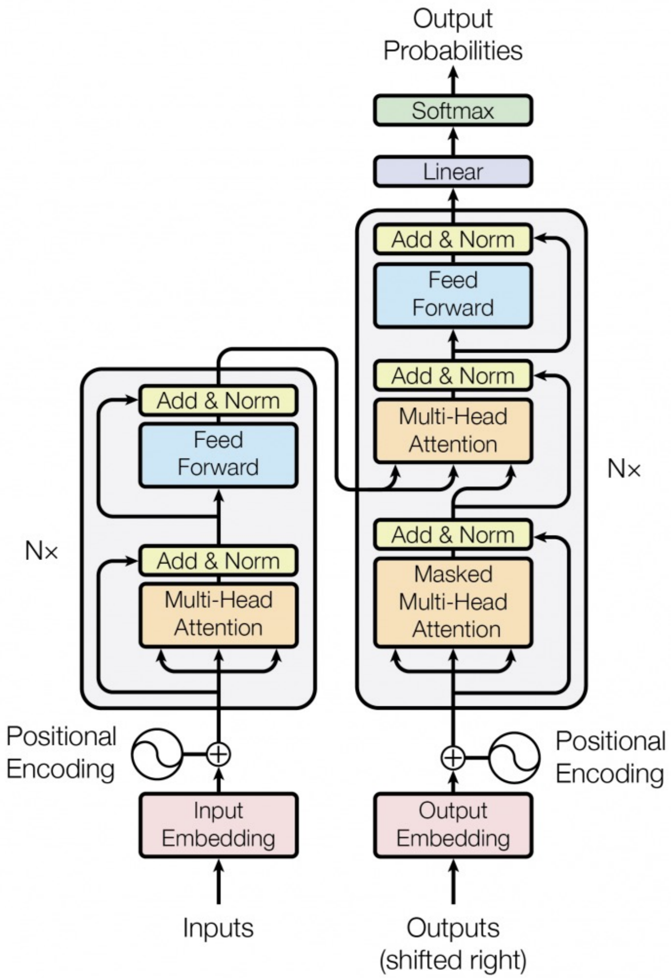
$$softmax:\ a_i = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

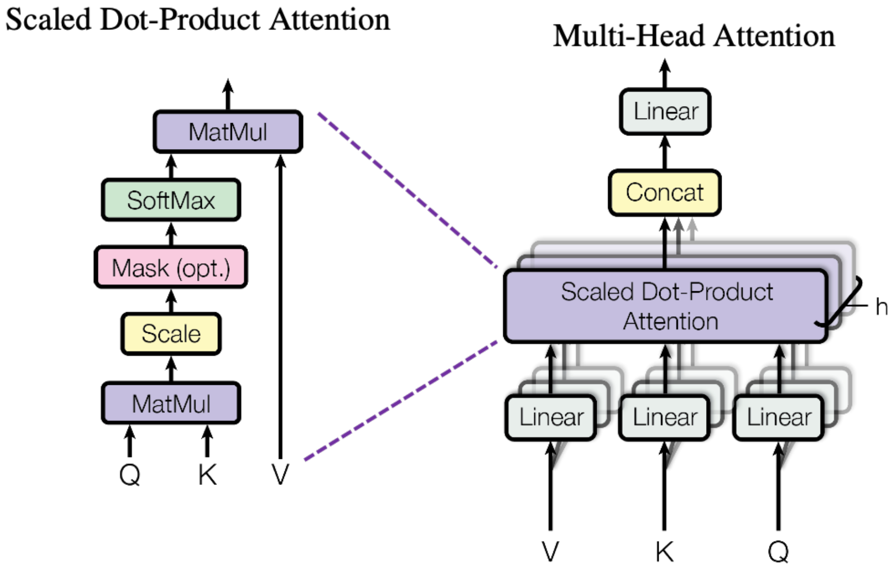
$$s_i = similarity(k_i, q) = \begin{cases} q^T k_i \\ q^T k_i/\sqrt{d} \\ q^T W k_i \\ \vdots \end{cases}$$

(d is the dimensionality of k)

Credit: STAT940, Prof. Ali Ghodsi, University of Waterloo

Background: Sequence Modeling with Transformers





Credit: https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/

Background: Transformers

Original [20]

$$z_i = \sum_{j=1}^{n} softmax\left(\{< q_i, k_{j'} >\}_{j'=1}^{n}\right).v_j$$

"The *cat* drank the milk because **it** was hungry."

GPT [21]

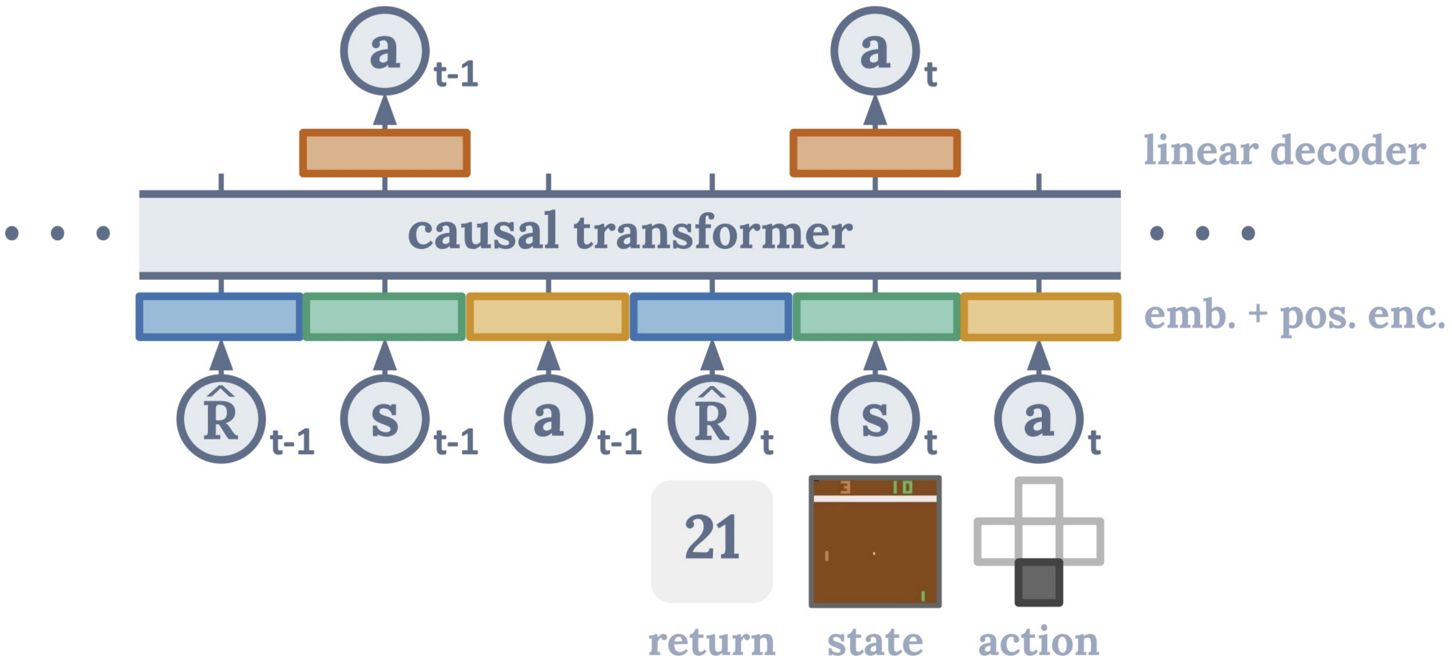
$$z_i = \sum_{j=1}^{i} softmax\left(\{< q_i, k_{j'} >\}_{j'=1}^{i}\right).v_j$$

"The *cat* drank the milk because **it** was hungry."

METHODOLOGY

Methodology: Decision Transformer [28] Overview

- Upside-down RL [13]
- Model-Free RL
- Offline RL
- Sequence Modelling using GPT
- Implicit Credit Assignment ✓
- No Distribution Shift ✓
- No expert demonstrations ✓
- Match or Exceed S.O.A. ✓



Methodology: Input Setup

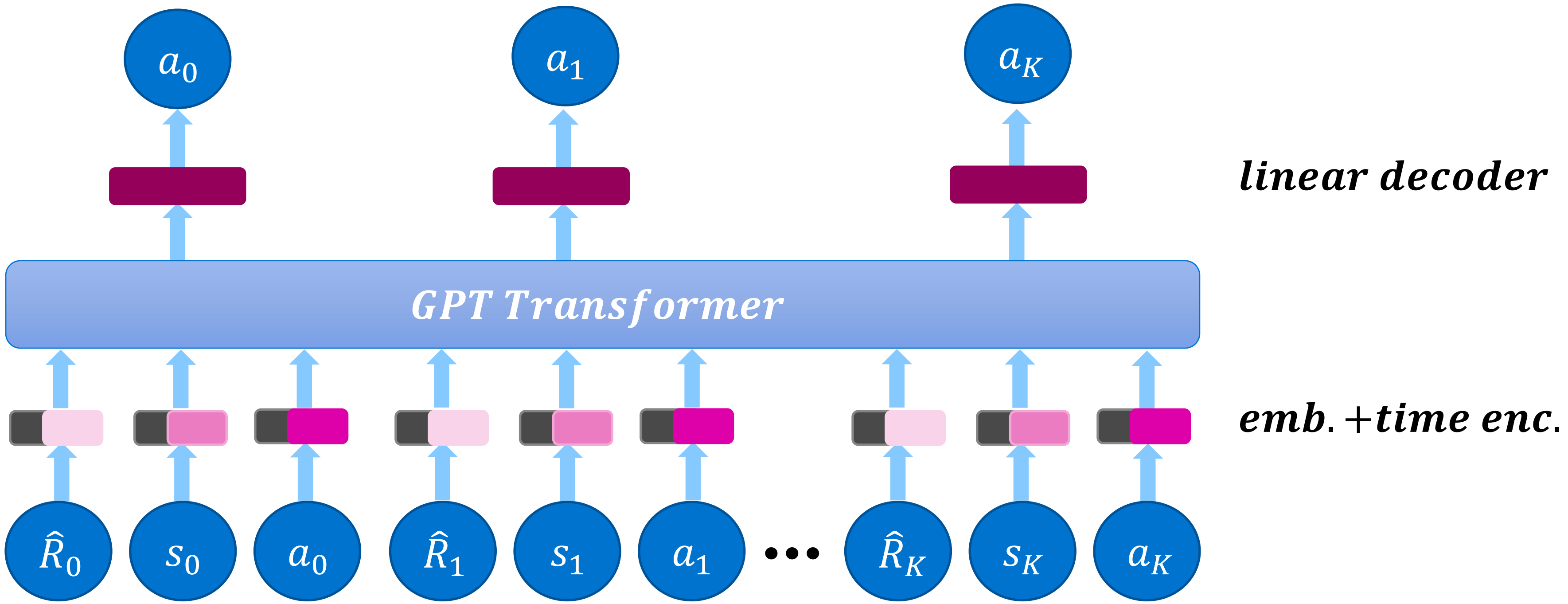
$$\tau = (r_0, s_0, a_0, r_1, s_1, a_1, \dots \dots, r_T, s_T, a_T)$$

$$\tau = (\hat{R}_0, s_0, a_0, \hat{R}_1, s_1, a_1, \dots \dots, \hat{R}_T, s_T, a_T)$$

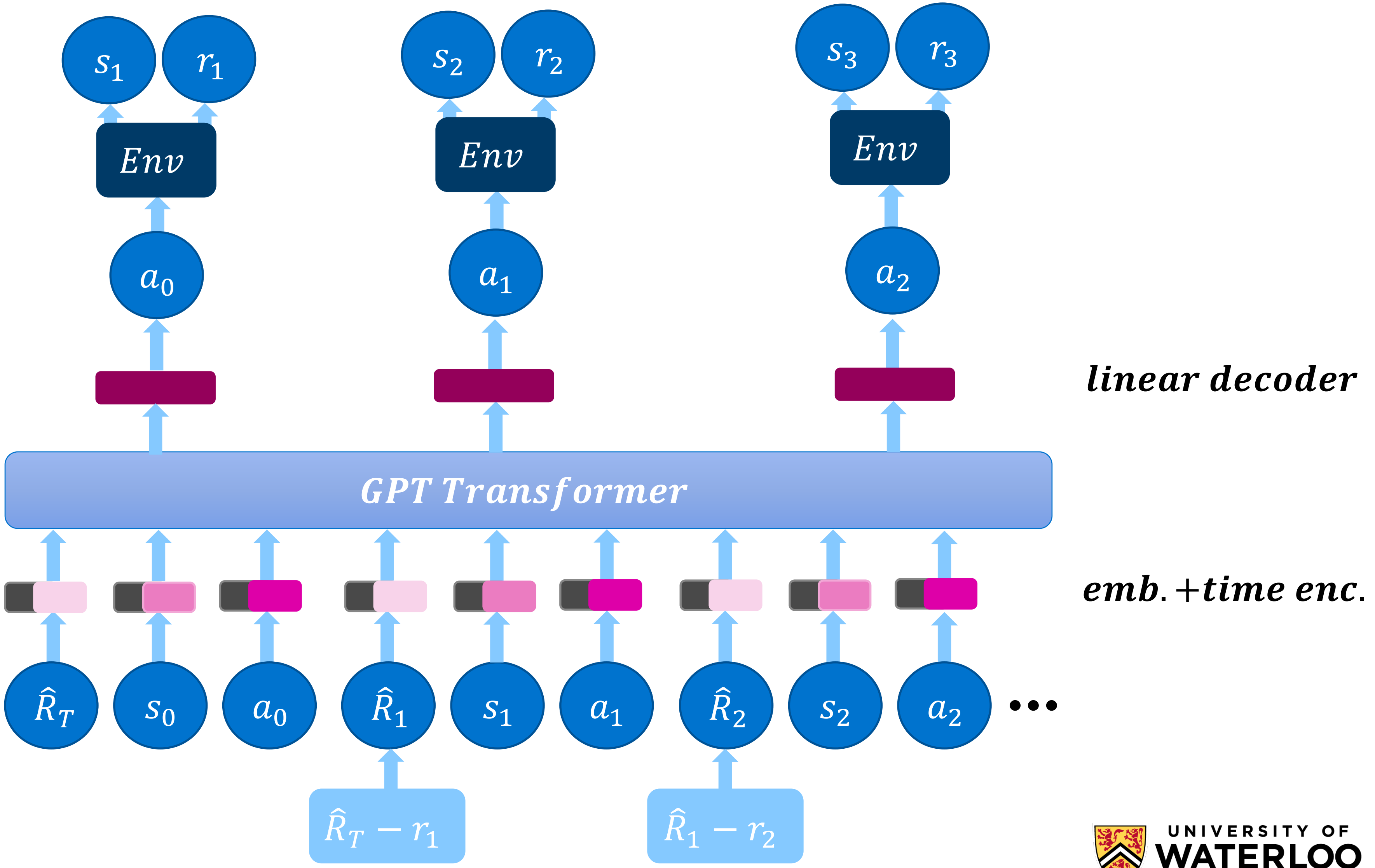
$$\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$$

Rewards-to-go

Methodology: Training Pipeline



Methodology: Inference Pipeline



Methodology: Psuedo-Code

Algorithm 1 Decision Transformer Pseudocode (for continuous actions)

```
# R, s, a, t: returns-to-go, states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_R: linear embedding layers
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer

# main model
def DecisionTransformer(R, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t)  # per-timestep (note: not per-token)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    R_embedding = embed_R(R) + pos_embedding

    # interleave tokens as (R_1, s_1, a_1, ..., R_K, s_K)
    input_embeds = stack(R_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
for (R, s, a, t) in dataloader:  # dims: (batch_size, K, dim)
    a_preds = DecisionTransformer(R, s, a, t)
    loss = mean((a_preds - a)**2)  # L2 loss for continuous actions
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
target_return = 1  # for instance, expert-level return
R, s, a, t, done = [target_return], [env.reset()], [], [1], False
while not done:  # autoregressive generation/sampling
    # sample next action
    action = DecisionTransformer(R, s, a, t)[-1]  # for cts actions
    new_s, r, done, _ = env.step(action)

    # append new tokens to sequence
    R = R + [R[-1] - r]  # decrement returns-to-go with reward
    s, a, t = s + [new_s], a + [action], t + [len(R)]
    R, s, a, t = R[-K:], ...  # only keep context length of K
```

EXPERIMENTS & RESULTS

Experiments: Atari Benchmark

Baselines

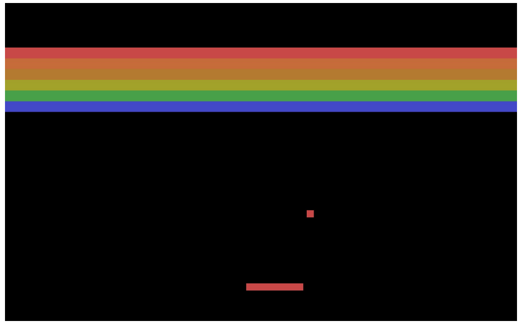
- CQL [22]
- REM [23]
- QE-DQN [24]
- BC (New)

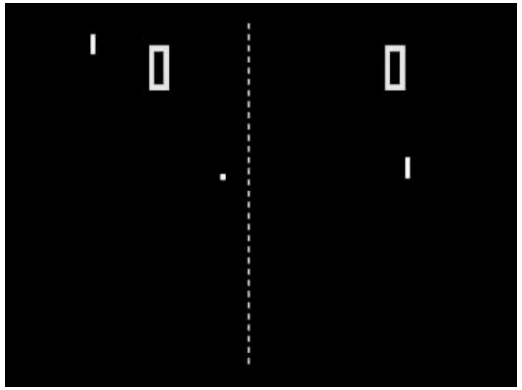
Games

- Breakout
- Qbert
- Pong (K=50)
- Seaquest

Challenges

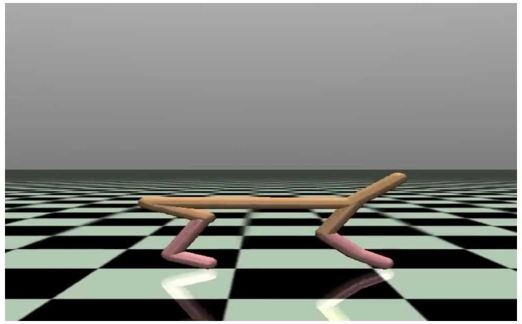
- Visual Inputs
- Long-term credit assignment

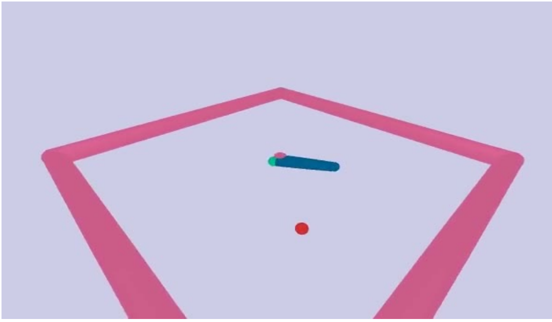


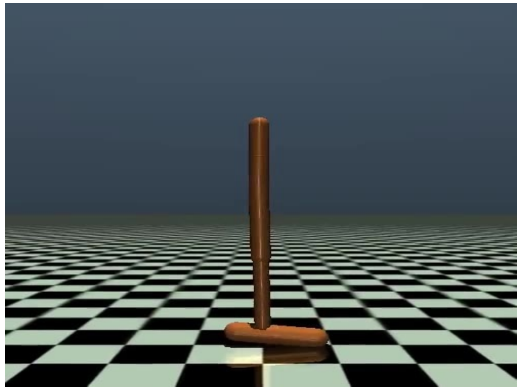


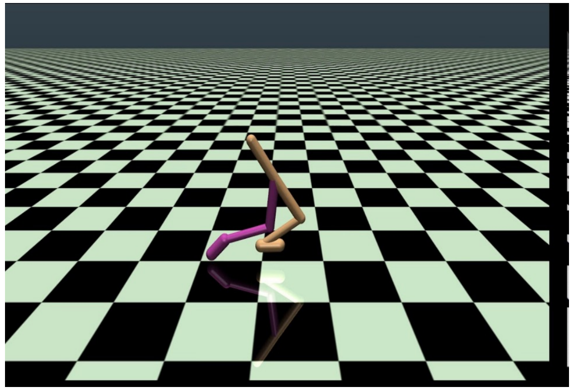
Experiments: D4RL [3] Benchmark

Baselines	Games	Dataset Settings
• CQL [22] • BEAR [25] • BRAC [26] • AWR [5] • BC (New)	• HalfCheetah • Hopper • Walker • Reacher (New)	• Medium • Medium-Replay • Medium-Expert









Results: Atari Benchmark

Game	DT (Ours)	CQL	QR-DQN	REM	BC
Breakout	$\mathbf{267.5 \pm 97.5}$	211.1	17.1	8.9	138.9 ± 61.7
Qbert	15.4 ± 11.4	**104.2**	0.0	0.0	17.3 ± 14.7
Pong	106.1 ± 8.1	**111.9**	18.0	0.5	85.2 ± 20.0
Seaquest	$\mathbf{2.5 \pm 0.4}$	1.7	0.4	0.7	2.1 ± 0.3

Results: D4RL [3] Benchmark

Dataset	Environment	DT (Ours)	CQL	BEAR	BRAC-v	AWR	BC
Medium-Expert	HalfCheetah	**86.8 ± 1.3**	62.4	53.4	41.9	52.7	59.9
Medium-Expert	Hopper	107.6 ± 1.8	**111.0**	96.3	0.8	27.1	79.6
Medium-Expert	Walker	**108.1 ± 0.2**	98.7	40.1	81.6	53.8	36.6
Medium-Expert	Reacher	**89.1 ± 1.3**	30.6	-	-	-	73.3
Medium	HalfCheetah	42.6 ± 0.1	44.4	41.7	**46.3**	37.4	43.1
Medium	Hopper	**67.6 ± 1.0**	58.0	52.1	31.1	35.9	63.9
Medium	Walker	74.0 ± 1.4	79.2	59.1	**81.1**	17.4	77.3
Medium	Reacher	**51.2 ± 3.4**	26.0	-	-	-	**48.9**
Medium-Replay	HalfCheetah	36.6 ± 0.8	46.2	38.6	**47.7**	40.3	4.3
Medium-Replay	Hopper	**82.7 ± 7.0**	48.6	33.7	0.6	28.4	27.6
Medium-Replay	Walker	**66.6 ± 3.0**	26.7	19.2	0.9	15.5	36.9
Medium-Replay	Reacher	**18.0 ± 2.4**	**19.0**	-	-	-	5.4
Average (Without Reacher)		**74.7**	63.9	48.2	36.9	34.3	46.4
Average (All Settings)		**69.2**	54.2	-	-	-	47.7

DISCUSSION

Q1: Does Decision Transformer perform behavior cloning on a subset of the data?

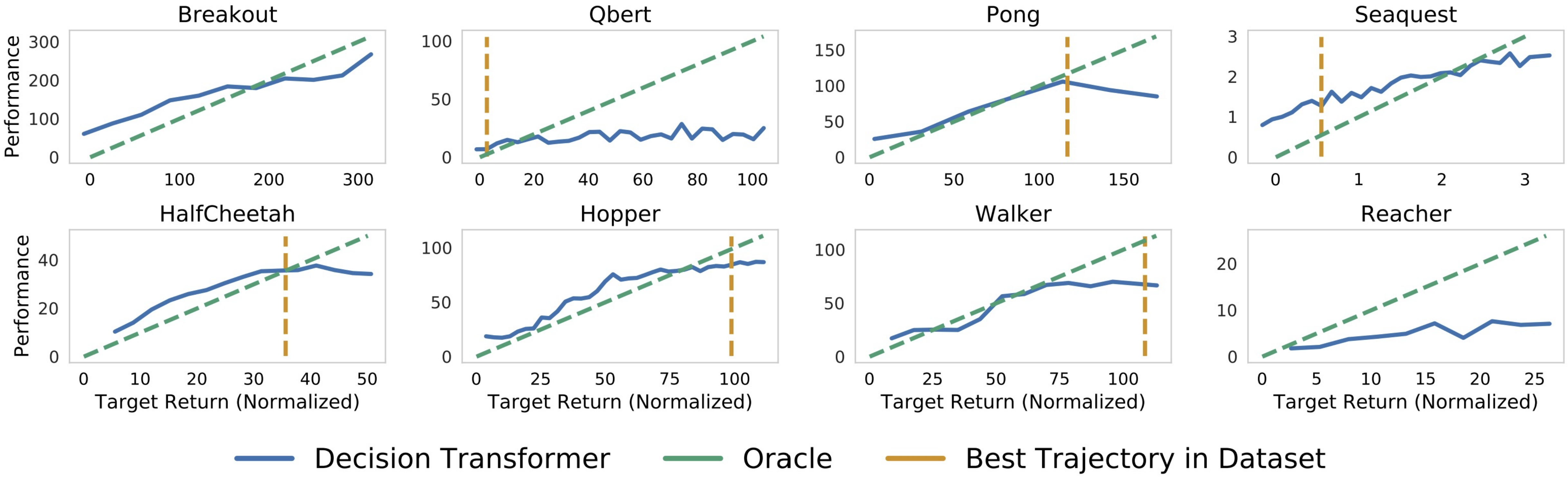
Large Dataset

Dataset	Environment	DT (Ours)	10%BC	25%BC	40%BC	100%BC	CQL
Medium	HalfCheetah	42.6 ± 0.1	42.9	43.0	43.1	43.1	**44.4**
Medium	Hopper	$\mathbf{67.6 \pm 1.0}$	65.9	65.2	65.3	63.9	58.0
Medium	Walker	74.0 ± 1.4	78.8	**80.9**	78.8	77.3	79.2
Medium	Reacher	51.2 ± 3.4	51.0	48.9	58.2	**58.4**	26.0
Medium-Replay	HalfCheetah	36.6 ± 0.8	40.8	40.9	41.1	4.3	**46.2**
Medium-Replay	Hopper	$\mathbf{82.7 \pm 7.0}$	70.6	58.6	31.0	27.6	48.6
Medium-Replay	Walker	66.6 ± 3.0	**70.4**	67.8	67.2	36.9	26.7
Medium-Replay	Reacher	18.0 ± 2.4	**33.1**	16.2	10.7	5.4	19.0
Average		56.1	**56.7**	52.7	49.4	39.5	43.5

Small Dataset

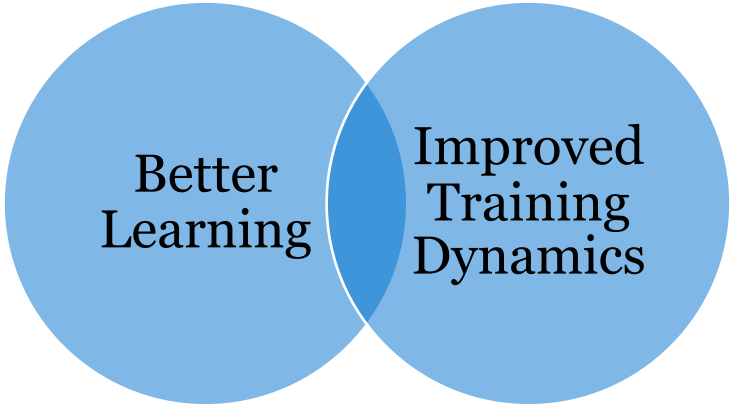
Game	DT (Ours)	10%BC	25%BC	40%BC	100%BC
Breakout	$\mathbf{267.5 \pm 97.5}$	28.5 ± 8.2	73.5 ± 6.4	108.2 ± 67.5	138.9 ± 61.7
Qbert	15.4 ± 11.4	6.6 ± 1.7	16.0 ± 13.8	11.8 ± 5.8	$\mathbf{17.3 \pm 14.7}$
Pong	$\mathbf{106.1 \pm 8.1}$	2.5 ± 0.2	13.3 ± 2.7	72.7 ± 13.3	85.2 ± 20.0
Seaquest	$\mathbf{2.5 \pm 0.4}$	1.1 ± 0.2	1.1 ± 0.2	1.6 ± 0.4	2.1 ± 0.3

Q2: How well does Decision Transformer model the distribution of returns?

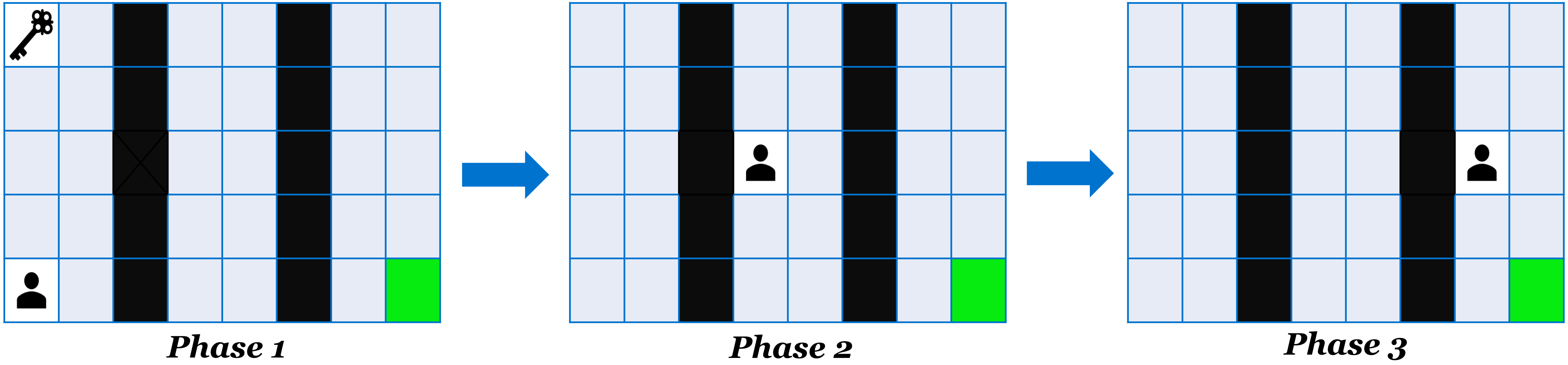


Q3: What is the benefit of using a longer context length?

Game	DT (Ours)	DT with no context ($K = 1$)
Breakout	$\mathbf{267.5 \pm 97.5}$	73.9 ± 10
Qbert	$\mathbf{15.1 \pm 11.4}$	13.6 ± 11.3
Pong	$\mathbf{106.1 \pm 8.1}$	2.5 ± 0.2
Seaquest	$\mathbf{2.5 \pm 0.4}$	0.6 ± 0.1



Q4: Does Decision Transformer perform effective long-term credit assignment?



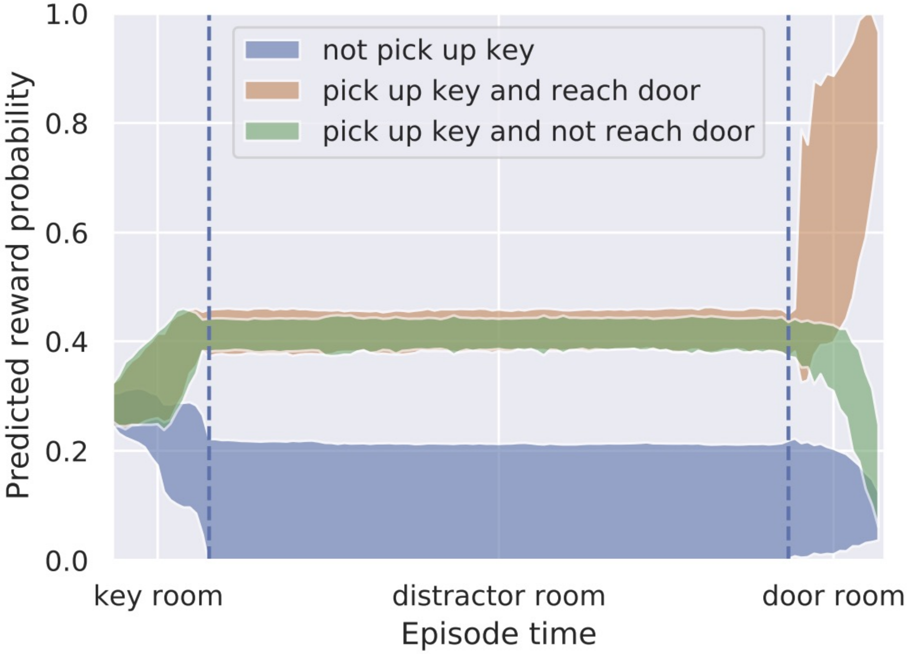
Dataset	DT (Ours)	CQL	BC	%BC	Random
1K Random Trajectories	**71.8%**	13.1%	1.4%	69.9%	3.1%
10K Random Trajectories	94.6%	13.3%	1.6%	**95.1%**	3.1%

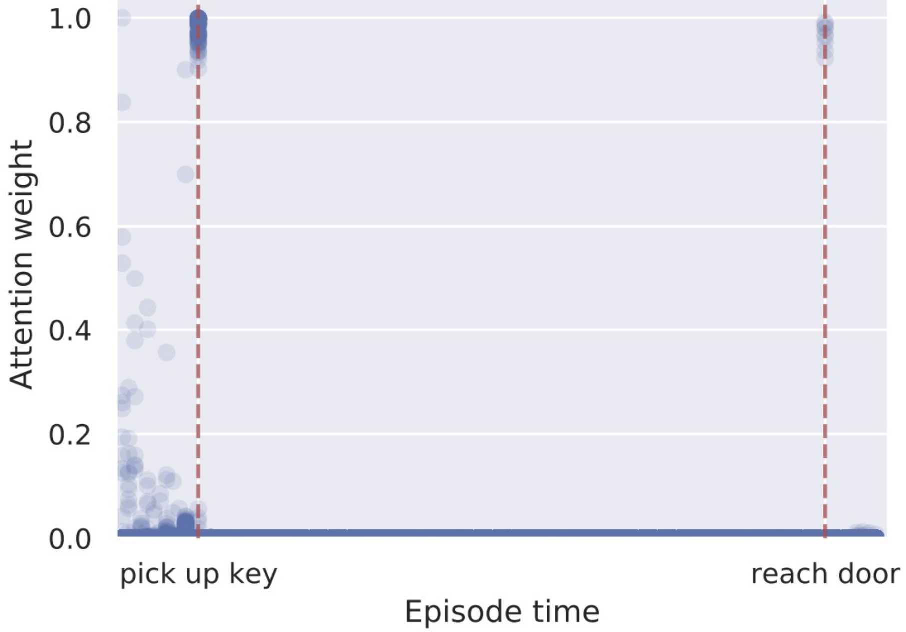
Q5: Can transformers be accurate critics in sparse reward settings?

Experiment Details:

Predict reward+action tokens

No initial returns-to-go





Q6: Does Decision Transformer perform well in sparse reward settings?

Experiment Details: → No rewards within trajectory → Final Cumulative reward at the end

Dataset	Environment	Delayed (Sparse) DT (Ours)	Delayed (Sparse) CQL	Agnostic BC	Agnostic %BC	Original (Dense) DT (Ours)	Original (Dense) CQL
Medium-Expert	Hopper	**107.3 ± 3.5**	9.0	59.9	102.6	107.6	111.0
Medium	Hopper	60.7 ± 4.5	5.2	63.9	**65.9**	67.6	58.0
Medium-Replay	Hopper	**78.5 ± 3.7**	2.0	27.6	70.6	82.7	48.6

Extra Observations

- No regularization or value pessimism needed
- Implicit representation of the value function
- Decision Transformer can benefit sample-efficient online regimes
- Can act as a strong model for behaviour generation

CONCLUSION

Conclusion

- Effective model-free supervised offline RL algorithm using sequence modelling.
- No reliance on any of the traditional RL concepts.
- Solves credit assignment and distribution shift problems seen in other RL algorithms.
- Match or surpass offline model-based RL state-of-the-art methods.

Future Work

- Use larger transformer models
- Conditioning on return distributions instead of discrete returns
- Model the state evolution using the transformer model to be an alternative for model-based RL.
- Understand the errors made by transformers for risks in real-world settings..

Limitations

Dependency on Context Length

Computational Time

Prior Knowledge on rewards

Loss of theoretical guarantees

References

[1] Christopher Watkins. Learning from delayed rewards. 01 1989

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

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

[4] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N.M., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. CoRR, abs/1509.02971.

[5] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019

[6] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In International Conference on Machine Learning, 2019.

[7] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Advances in Neural Information Processing Systems, 2019.

[8] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Advances in Neural Information Processing Systems, 2020.

[9] Oh, J., Y. Guo, S. Singh, and H. Lee 2018. Self-Imitation Learning. arXiv:1806.05635 [cs, stat].

[10] Arjona-Medina, J. A., M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter 2019. Rudder: Return decomposition for delayed rewards. In Advances in Neural Information Processing Systems, Pp. 13544–13555.

[11] Pomerleau, D.A. (1988). ALVINN: An Autonomous Land Vehicle in a Neural Network. NIPS.

References

[12] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jürgen ´ Schmidhuber. Training agents using upside-down reinforcement learning. arXiv preprint arXiv:1912.02877, 2019.

[13] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. arXiv preprint arXiv:1912.13465, 2019.

[14] Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. arXiv preprint arXiv:1912.06088, 2019.

[15] Keiran Paster, Sheila A McIlraith, and Jimmy Ba. Planning from pixels using inverse dynamics models. arXiv preprint arXiv:2012.02419, 2020.

[16] Johan Ferret, Raphaël Marinier, Matthieu Geist, and Olivier Pietquin. Self-attentional credit assignment for transfer in reinforcement learning. arXiv preprint arXiv:1907.08027, 2019.

[17] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Greg Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. arXiv preprint arXiv:1912.02503, 2019.

[18] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. arXiv preprint arXiv:1806.07857, 2018

[19] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. Nature communications, 10(1):1–12, 2019.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, 2017.

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

[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Advances in Neural Information Processing Systems, 2020.

[23] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In International Conference on Machine Learning, 2020.

[24] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Conference on Artificial Intelligence, 2018.

References

[25] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. arXiv preprint arXiv:1906.00949, 2019.

[26] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361, 2019.

[27] Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. arXiv preprint arXiv:2011.09464, 2020.

[28] 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. ArXiv, abs/2106.01345.

UNIVERSITY OF WATERLOO

FACULTY OF MATHEMATICS

https://arxiv.org/pdf/2106.01345.pdf