Decision Transformer: Reinforcement Learning via Sequence Modeling

NeurIPS 2021

Lili Chen[*,1], **Kevin Lu**[*,1], **Aravind Rajeswaran**[2], **Kimin Lee**[1],
Aditya Grover[2], **Michael Laskin**[1], **Pieter Abbeel**[1], **Aravind Srinivas**[†,1], **Igor Mordatch**[†,3]

[*]equal contribution [†]equal advising

[1]UC Berkeley [2]Facebook AI Research [3]Google Brain

{lilichen, kzl}@berkeley.edu

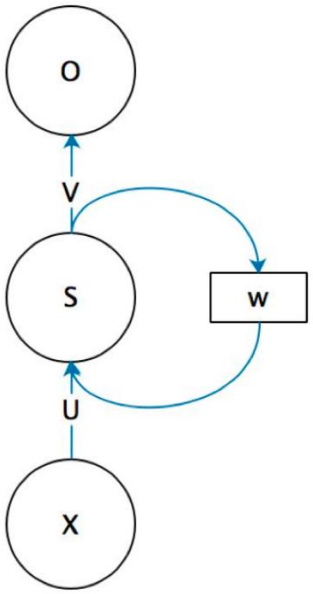
问题？

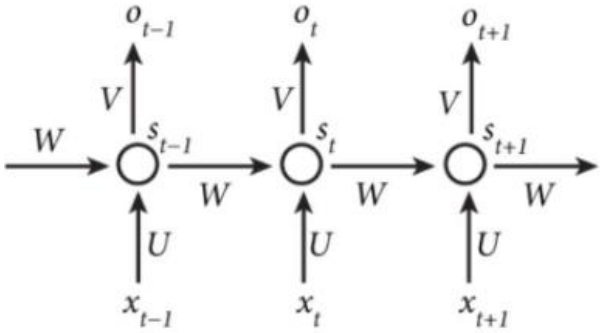
强化学习主要研究的是以序列（轨迹）为相关的一系列问题， 而现在所有的方法基于都是 **MDP + TD learning** 的方法，但是 **TDs** 容易导致 “Deadly Triad” 的产生， 所以能不能将 **RL** 问题直接看成一个序列建模问题？

Deadly Triad issue of TDs:

1. Bootstrapping
2. Off-policy learning
3. Function approximations

RNN

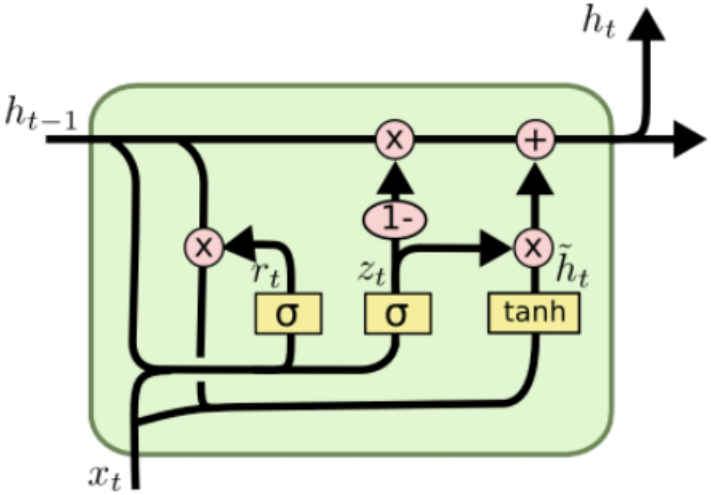




$$O_t = g(V \cdot S_t)$$

$$S_t = f(U \cdot X_t + W \cdot S_{t-1})$$

LSTM



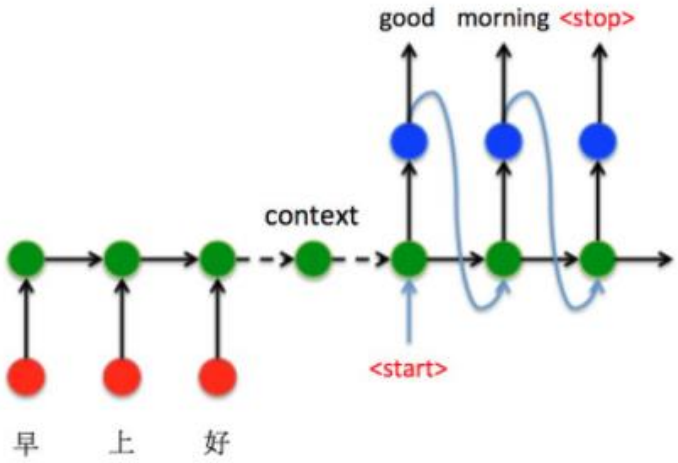
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Seq2Seq



Teacher Forcing: decoder阶段将groud-truth 作为下一步的输入。

好处：收敛快，好训练。
坏处：测试阶段没有gt，只能用Autoregressive方法，由于训练和预测的时候decode行为的不一致， 导致预测在训练和测试的时候是从不同的分布中推断出来的。

Autoregressive：decoder阶段将输出值作为下一步的输入。

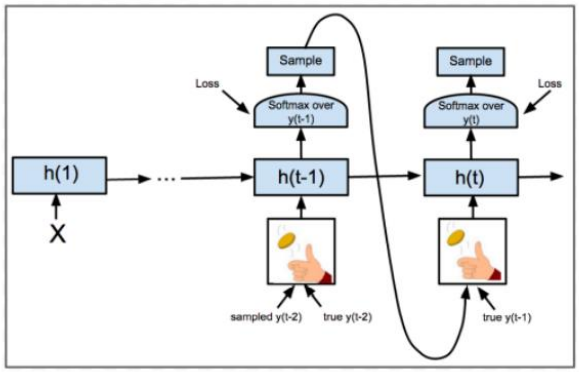


Figure 1: Illustration of the Scheduled Sampling approach, where one flips a coin at every time step to decide to use the true previous token or one sampled from the model itself.

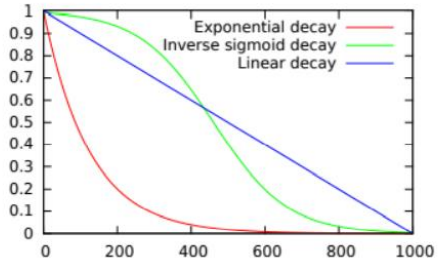
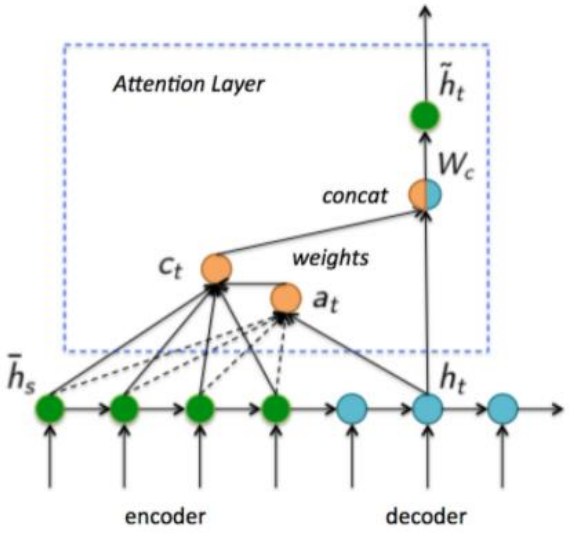


Figure 2: Examples of decay schedules.

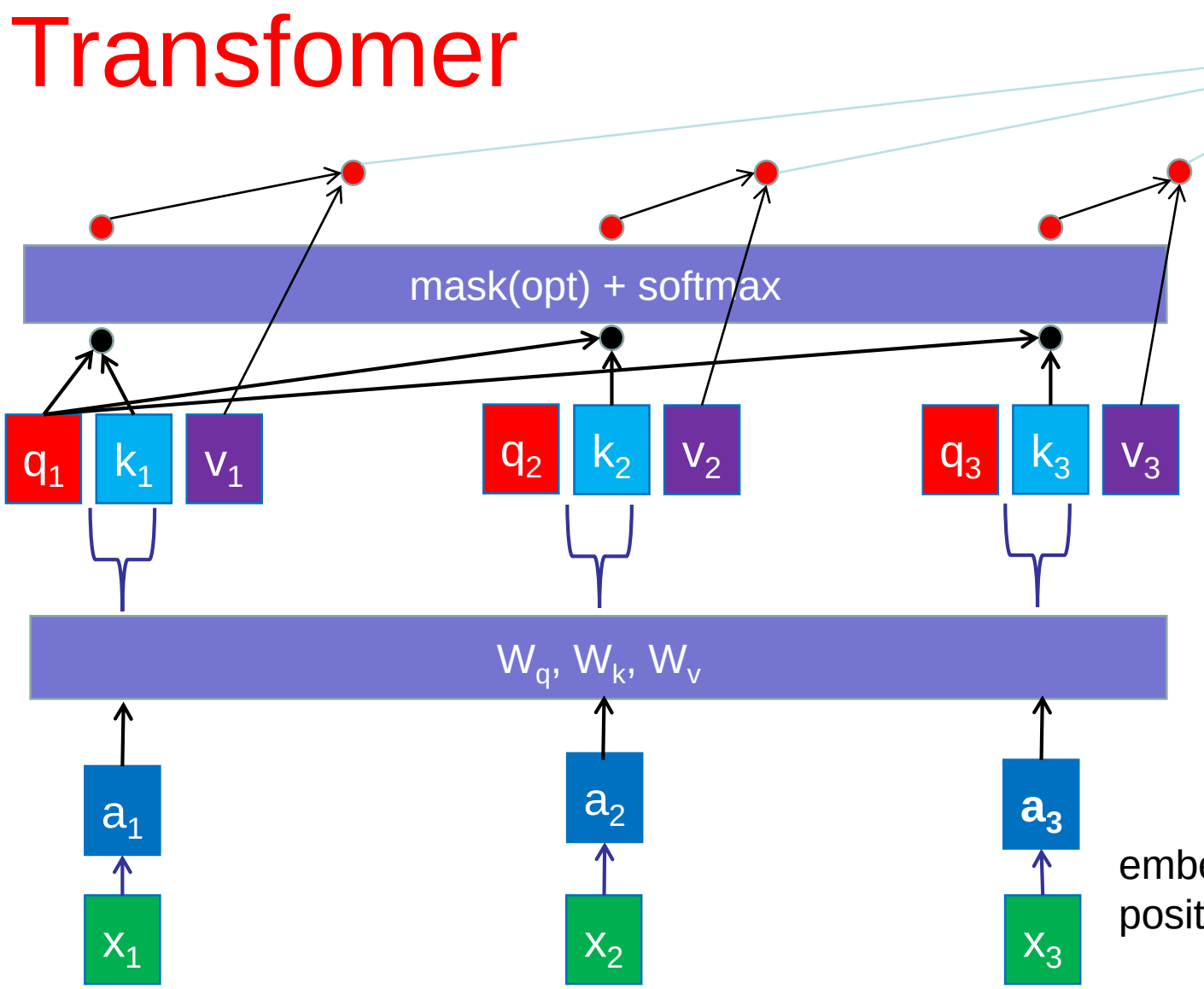
Attention机制

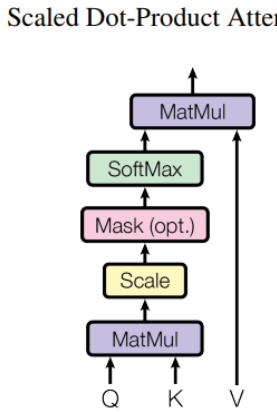


$$\alpha_{ts} = \frac{\exp(score(h_t, \bar{h}s))}{\sum s' = 1^{S} \exp(score(h_t, \bar{h}s'))} \qquad [\text{Attention weights}]$$

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} h_t^T \bar{h}_s & \text{Dot} \\ h_t^T W_a \bar{h}_s & \text{General} \\ v_a^T \tanh(W_a \cdot \text{concat}(h_t, \bar{h}_s)) & \text{Concat} \end{cases}$$

Transfomer

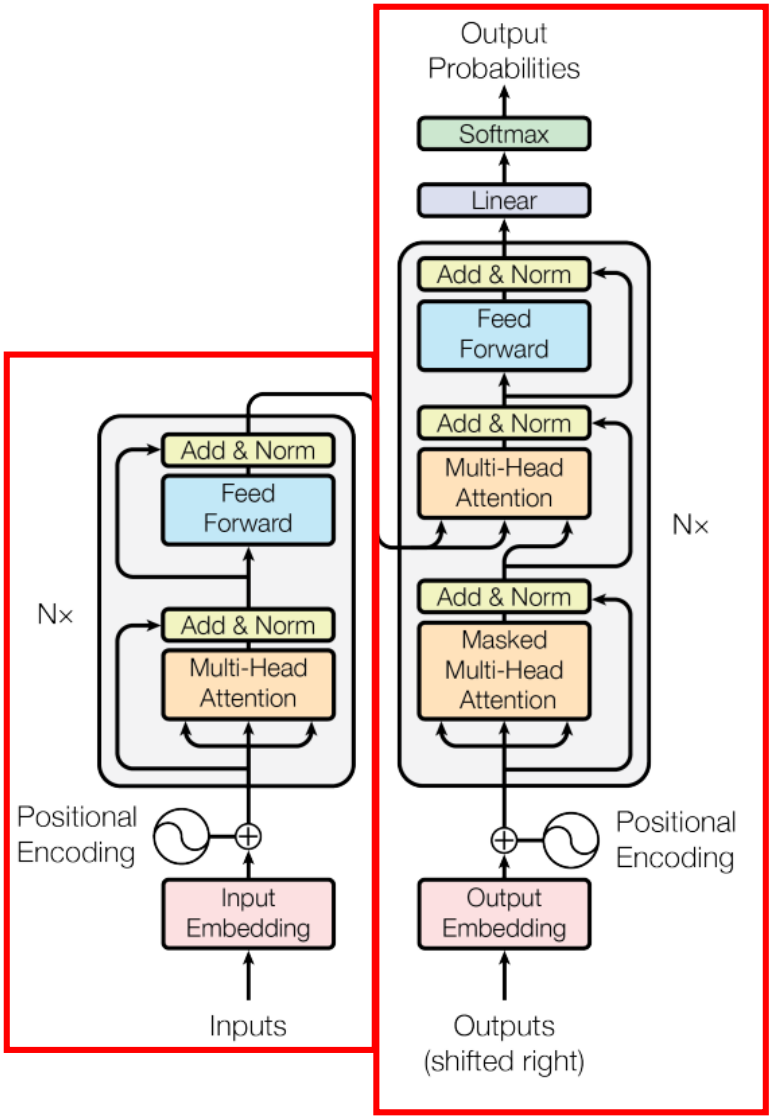




embedding +
positional encoding

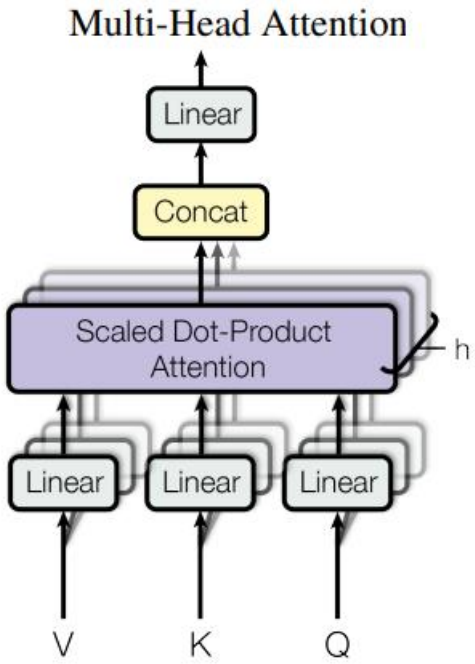
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Transfomer



encoder

decoder



Decision Transformer

returns-to-go $\hat{R}_t = \sum_{t'=t}^{T} r_t$

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

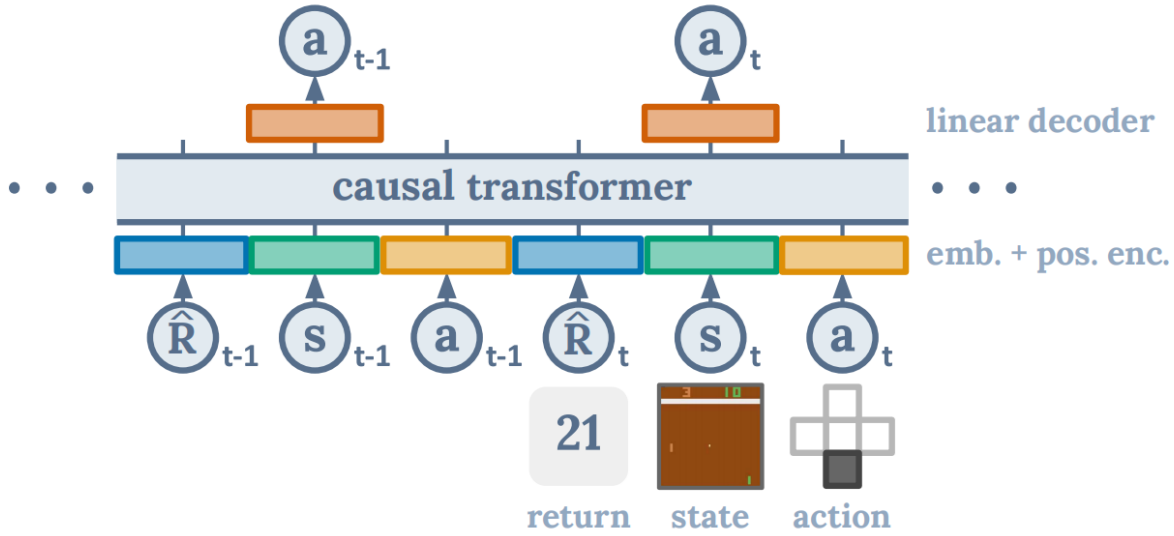


Figure 1: Decision Transformer architecture[1]. States, actions, and returns are fed into modality-specific linear embeddings and a positional episodic timestep encoding is added. Tokens are fed into a GPT architecture which predicts actions autoregressively using a causal self-attention mask.

好处： transfomer 天然自带 credit assignment

Algorithm

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
```

Result

Atari

Game	DT (Ours)	CQL	QR-DQN	REM	BC
Breakout	**267.5 ± 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	**2.5 ± 0.4**	1.7	0.4	0.7	2.1 ± 0.3

Table 1: Gamer-normalized scores for the 1% DQN-replay Atari dataset. We report the mean and variance across 3 seeds. Best mean scores are highlighted in bold. Decision Transformer (DT) performs comparably to CQL on 3 out of 4 games, and outperforms other baselines in most games.

OpenAI Gym

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

Table 2: Results for D4RL datasets[3]. We report the mean and variance for three seeds. Decision Transformer (DT) outperforms conventional RL algorithms on almost all tasks.

Discussion

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

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

Table 3: Comparison between Decision Transformer (DT) and Percentile Behavior Cloning (%BC).

Discussion

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

Table 5: Ablation on context length. Decision Transformer (DT) performs better when using a longer context length ($K = 50$ for Pong, $K = 30$ for others).

Discussion

Can transformers be accurate critics in sparse reward settings?

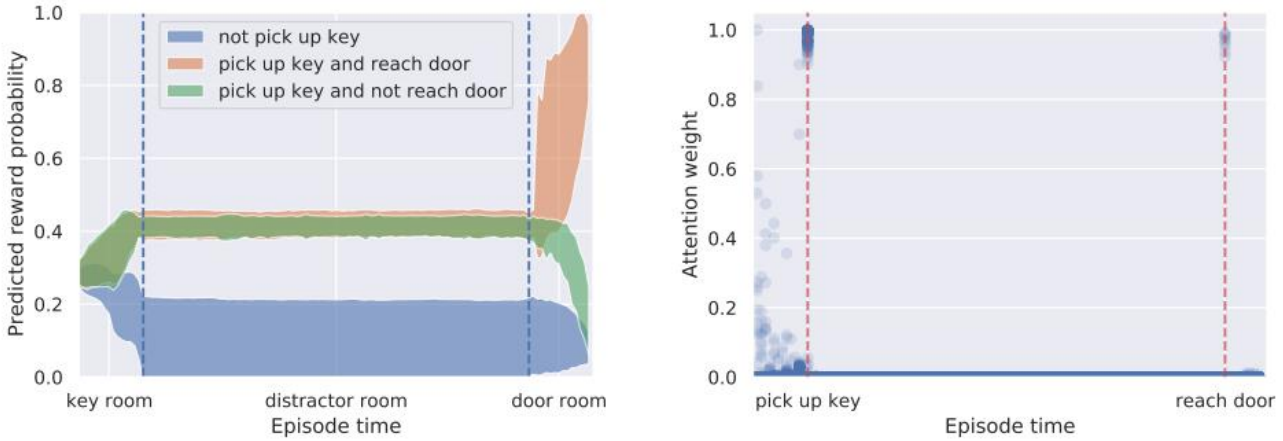


Figure 5: **Left:** Averages of running return probabilities predicted by the transformer model for three types of episode outcomes. **Right:** Transformer attention weights from all timesteps superimposed for a particular successful episode. The model attends to steps near pivotal events in the episode, such as picking up the key and reaching the door.

Trajectory Transformers

$$\tau = \{\mathbf{s}_t^0, \mathbf{s}_t^1, \ldots, \mathbf{s}_t^{N-1}, \mathbf{a}_t^0, \mathbf{a}_t^1, \ldots, \mathbf{a}_t^{M-1}, r_t\}_{t=0}^{T-1}.$$

$$\mathcal{L}(\bar{\tau}) = \sum_{t=0}^{T-1} \Big(\sum_{i=0}^{N-1} \log P_\theta\big(\bar{\mathbf{s}}_t^i \mid \bar{\mathbf{s}}_t^{<i}, \bar{\tau}_{<t}\big) + \sum_{j=0}^{M-1} \log P_\theta\big(\bar{\mathbf{a}}_t^j \mid \bar{\mathbf{a}}_t^{<j}, \bar{\mathbf{s}}_t, \bar{\tau}_{<t}\big) + \log P_\theta\big(\bar{r}_t \mid \bar{\mathbf{a}}_t, \bar{\mathbf{s}}_t, \bar{\tau}_{<t}\big) \Big)$$