Decision Transformer: Reinforcement Learning via Sequence Modeling

Deep Learning – Final Project Report

Ben Kabongo ben.kabongo_buzangu@ens-paris-saclay.fr
Master MVA, ENS Paris-Saclay

January 2024

Abstract

The paper presents a new framework that abstracts reinforcement learning as a sequence modeling problem. This takes advantage of the simplicity and scalability of the Transformer architecture and associated advances in language modeling.
Decision Transformer is an architecture that transforms the reinforcement learning problem into conditional sequence modeling. Unlike previous reinforcement learning approaches that fit value functions or compute policy gradients, Decision Transformer simply produces optimal actions by relying on a causally masked Transformer.
Decision Transformer meets or exceeds the performance of state-of-the-art model-free offline reinforcement learning databases on Atari, OpenAI Gym and Key-to-Door tasks [12].
Our experiments on Decision Transformer focus on simple Markov Decision Processes and our implementation is available here: https://github.com/BenKabongo25/mva-decision-transformer-project.

1 Introduction

Reinforcement learning is a branch of machine learning in which an agent learns to make decisions by interacting with an environment. The agent receives rewards or sanctions based on the actions it takes, allowing it to adjust its behaviour to maximise rewards in the long term [63]. The aim of reinforcement learning is to find the optimal policy, the decision-making behaviour that maximises the sum of the rewards. Various conventional algorithms can be used to find optimal policies, based on value functions or by calculating policy gradients. Some of these algorithms are: TD-Learning [63], Q-Learning [65], Deep Q-Network [46] and their various variants. The history of interactions between the agent and the environment over time defines a trajectory. This trajectory can be likened to a sequence. It may therefore be appropriate to approach reinforcement learning problems as sequence modeling problems.

Transformers are the state-of-the-art in sequence modeling tasks. The performance of these models for language modeling surpasses that of recurrent neural networks. Given the diversity of successful applications of these models, the authors of the paper seek to examine their application to sequential decision-making problems formalised as reinforcement learning. The authors therefore seek to investigate whether generative trajectory modeling can be used as a replacement for conventional reinforcement learning algorithms. Previous work [49] [71] has used transformers as an architectural choice for components in traditional reinforcement learning algorithms.

Thus, instead of learning a policy using conventional reinforcement learning algorithms, the paper proposes to train models of transformers based on experiences collected using a sequence modeling objective. This circumvents the need for priming for long-term credits known to destabilise the reinforcement learning [63]. There is also no need to discount future rewards, as is typically the case in temporal difference learning, which can induce undesirable short-sighted behaviour. Unlike Bellman backups, which propagate rewards slowly and are subject to distracting signals [31], transformers can perform crediting directly

through self-attention. This ensures that transformers perform effectively in the presence of sparse or distracting rewards. A modeling approach using transformers allows better generalisation and transfer, as they can model a wide distribution of behaviours [55].

Experiments with such an approach are then carried out in an offline reinforcement learning framework, where the model has to learn policies from sub-optimal data from different trajectories. This task is natural when training is aimed at sequence modeling, but remains difficult due to error propagation and overestimation of values [41]. By training an autoregressive model on sequences of states, actions and returns, policy sampling is reduced to autoregressive generative modeling.

2 Problem Definition

2.1 Markov Decision Process and Reinforcement learning

Consider an agent immersed in an environment with which it interacts. Let be:

1. **A state**: the current perception of the environment from the agent's point of view. Let us denote $\mathcal{S}$, the set of states, finite or continuous.
2. **An action**: the different ways in which the agent interacts with the environment. These are the decisions taken by the agent at a given moment. Let's denote $\mathcal{A}$ the set of actions.
3. **Transition function** $\mathcal{P}$: which gives the next state for an action taken from a given state. The transition function can be deterministic ($\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$) or probabilistic ($\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$).
4. **Reward function** $\mathcal{R}$: which gives the associated reward for an action performed from a given state. This is a quantitative evaluation of the decision taken by the agent. The reward can be positive or negative. We can have a deterministic reward function that depends on the current state s and the action performed a given by $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

A **Markov decision process** (MDP) is a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ [7].

We call **policy**, denoted π, a function which gives the action to choose for a given state. This can be a deterministic policy given by $\pi : \mathcal{S} \to \mathcal{A}$ or a probabilistic policy given by $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The agent's objective is to find the optimal policy.

By following a given policy, with the triplet (s_t, a_t, r_t) describing the state at time t, the action chosen at time t and the reward obtained at time t, the agent defines a **trajectory**, given by the set of triplets over time: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_T, a_T, r_T)$. Let $\gamma \in [0, 1]$ be a constant used to weight future rewards, the global reward associated with a trajectory is: $R_{t0} = \sum_{t=0}^{T-t0} \gamma^t r_{t_0+t}$.

The objective in reinforcement learning is to find the optimal decision-making behaviour which maximises the sum of rewards over time. The **value functions** give an expectation of cumulative future rewards [63]. The state value function, $V^\pi : \mathcal{S} \to \mathbb{R}$, gives the expectation of the agent's cumulative future rewards if he were to start from a given state: $V^\pi(s) = \mathbb{E}_\pi(R_{t0}|s_{t0} = s)$. The state-action value function, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, gives the expectation of future earnings for a given state-action pair: $Q^\pi(s, a) = \mathbb{E}_\pi(R_{t0}|s_{t0} = s, a_{t0} = a)$. When we have complete knowledge of the possible transitions and rewards, the optimal policy can be found by dynamic programming. When the model of the environment is not completely known, various algorithms can still be used to find optimal policies.

A distinction is made between online and offline reinforcement learning. In an online framework, the optimal policy is learned during the various interactions between the agent and the environment. In offline reinforcement learning, instead of obtaining data through interactions with the environment, we only have access to a limited, fixed dataset of arbitrary policy trajectories. Decision Transformer uses the offline learning mode.

2.2 Transformers

Transformers [64] are a neural network architecture originally designed for sequence processing in natural language processing. The architecture was then used in other fields such as computer vision and, increasingly, reinforcement learning. Transformers are completely connected networks and, compared with recurrent neural networks, have the advantage of being able to process the sequence in the desired order (enabling parallel processing) and being resistant to gradient evaporation problems. As a result, transformers quickly became very popular for the various problems associated with natural language processing.

These models consist of stacked self-attention layers with residual connections. Each self-attention layer receives n embeddings $\{x_i\}_{i=1}^n$ corresponding to unique input tokens, and outputs n embeddings $\{z_i\}_{i=1}^n$. The i-th token is mapped via linear transformations to a key k_i, query q_i, and value v_i. The i-th output of the self-attention layer is given by weighting the values v_j by the normalized dot product between the query q_i and other keys k_j: $z_j = \sum_{j=1}^n \text{softmax}(\{\langle q_i, k_{j'} \rangle\}_{j'=1}^n)_j \cdot v_j$

2.3 Reinforcement Learning as sequence modeling

The authors of the paper present Decision Transformer, which uses the GPT [53] architecture to model trajectories in an autoregressive manner. The experiments carried out in the paper show that sequence modeling can be used to optimise policies without using dynamic programming, achieving or exceeding the performance of the most advanced classical offline reinforcement learning algorithms. In addition, Decision Transformer is able to outperform basic reinforcement learning algorithms in tasks where long-term crediting is required.

The experiments performed in the paper were carried out on offline reinforcement learning benchmarks in the Atari [6], OpenAI Gym [8] and Key-to-Door [45] environments.

3 Related Work

Attention and transformer models. Despite the success of Transformers [64] in various natural language processing [16] [53] and computer vision tasks [11] [17], they have received very little attention in reinforcement learning. Unlike Decision Transformer, which uses a transformer architecture to abstract reinforcement learning tasks into sequence modeling tasks, previous work [71] [57] [49] incorporates transformers, still using actor-critic algorithms for optimization. Other work [14] [1] in imitation learning has focused on the use of tranformers in architectures using recurrent neural networks.

Conditional language generation. Class conditional language models can be used to learn disciminators to guide generation [25] [15] [29] [37]. Several works [21] [30] [54] [69] [72] have explored the training or fine-tuning of models for controllable text generation and other works have studied guided generation of images [34] and language [25] [66]. However, these approaches mostly assume constant classes, while in reinforcement learning the reward signal varies over time.

Supervised learning in reinforcement learning settings. Q-learning [65] [46] like some previous reinforcement learning methods, closely resembles statistical supervised learning, using iterative saves or likelihood-based methods. Recent work [61] [39] studies reverse reinforcement learning seeking to model behaviors with a supervised loss conditioned on the targeted performance. The difference with Decision Transformer is that the latter focuses more on sequence modeling rather than supervised learning. Sequence modeling allows behavior to be modeled even without access to reward and is known to scale well [9]. Another work [33] proposes Trajectory Transformer, which is similar to Decision Transformer but additionally uses state and feedback prediction, as well as discretization, which integrates model-based components. The results of the experiments in both works highlight the potential of sequence modeling as a generally applicable idea for reinforcement learning.

Offline reinforcement learning. In the paper, the authors compare Decision Transformer to other classic model-free reinforcement learning algorithms, because it is an approach that does not explicitly learn the dynamics of the model and is trained in a context of offline learning. Offline learning is sensitive to

distribution change. Previous works [40] [23] [38] [59] [36] [70] have addressed this problem in various ways. Other work explores learning a large distribution of behaviors from an offline dataset, either with likelihood-based approaches [3] [10] [52] [60] or by maximizing mutual information [19] [44] [58]. Decision Transformer is similar to likelihood-based approaches, which do not use iterative Bellman updates.

Credit assignment. By learning an architecture that decomposes the reward function in such a way that certain important states obtain the majority of the credits, many works [20] [28] [45] have studied better allocation of credits by association of states. Other work [4] [31] [42] [56] has specifically shown that such state associative architectures can perform better in delayed reward contexts. Decision Transformer does not explicitly learn a reward function or a critic. The Transformer architecture is capable of making it emerge naturally.

4 Method

4.1 Trajectory representation

Let $\hat{R}_t = \sum_{t=1}^{T} r_t$. The trajectories for the Decision Transformer model are represented by:

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

The authors justify this choice by the fact that it allows transformers to learn meaningful patterns and is able to conditionally generate actions at the time of testing. So it's preferable for the model to generate actions based on desired future returns, rather than past rewards. Therefore, instead of immediate rewards, the model is driven by returns-to-go.

4.2 Architecture

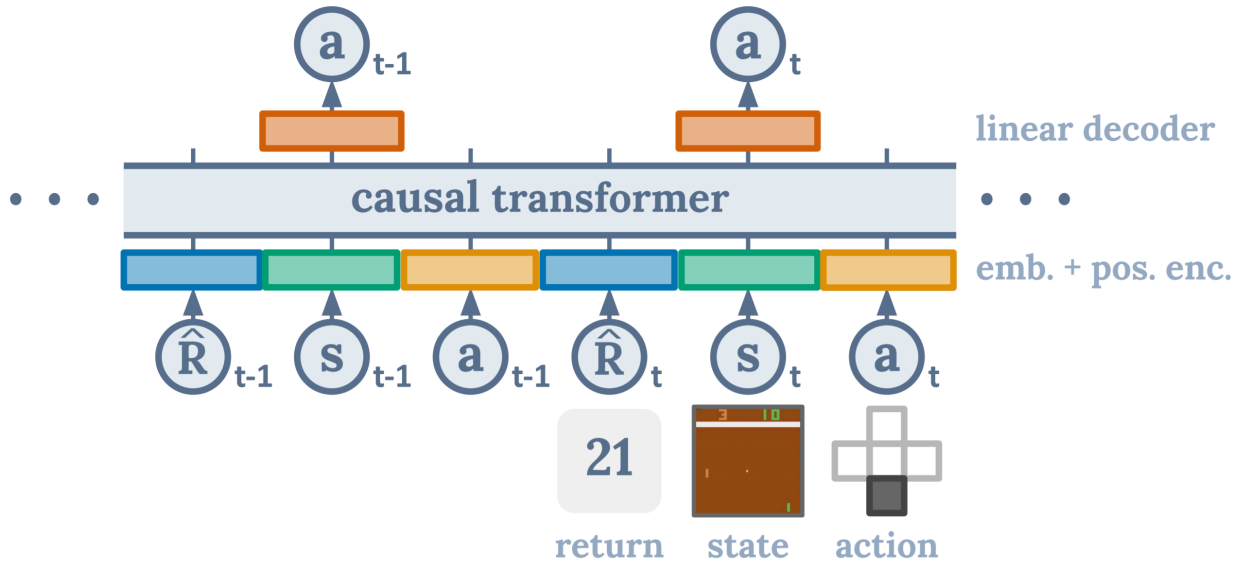

Figure 1: Decision Transformer architecture [12]

Decision Transformer takes input on the last K time steps, for a total of 3K tokens. For each time step, we have three modalities: return-to-go, state and action. For each modality, a linear layer projects the raw input data into the embedding dimension of the model. We thus obtain embeddings for all the tokens. Additionally, an embedding for each time step is learned and added to each corresponding token. A time step corresponds to three tokens for the three modalities. The tokens are then processed by a GPT model [53], which predicts future action tokens through autoregressive modeling.

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
```

Figure 2: Decision Transformer pseudo-code [12]

4.3 Training

The model is trained on an offline trajectory dataset, from which batches are sampled over K time steps. The model is trained on action prediction. Thus, we will use a cross-entropy loss for discrete actions and a mean square error for continuous actions.

5 Evaluation

5.1 Environnements and datasets

In the paper under review, experiments to evaluate Decision Transformer were carried out on Atari, OpenAI Gym and Key-to-Door tasks. These are fairly complex environments. Particularly for Atari, where actions are discrete, a state is represented by four images and then encoded with convolutional neural networks in the Decision Transformer dedicated to this type of environment. In addition, the training data is very large. Experimenting with these environments and the Decision Transformer model requires access to powerful computers with GPUs. However, during the course of this project, we had rather limited access to GPUs and powerful computers to carry out experiments in these same environments. We therefore chose to create simple environments for our experiments.

5.1.1 Environnements

For the creation of environments, we take up the notion of MDPs.

States and actions. We define MDPs with a discrete $\mathcal{S}$ set of states and a discrete $\mathcal{A}$ set of actions, so that the number of states and the number of actions are always known.

Transition function. The transition functions taken into account are of the form $\mathcal{P} : \mathcal{S} \to \mathcal{S}$ for deterministic transition functions based solely on states, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ for deterministic transition functions based on states and actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ for probabilistic transition functions. For the first two forms, we also define their probabilistic versions.

Reward function. The reward functions considered are in the form $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ for the deterministic case based on states only, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for the deterministic case based on states and actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ for the deterministic case based on state-action-state triplets and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R} \to [0, 1]$ for probabilistic cases. For the first three forms, we also define their probabilistic versions.

Termination function. We also define a binary termination function $\mathcal{T} : \mathcal{S} \to \{0, 1\}$ which indicates for each state whether it is terminal or not.

The environments can be used interactively, in particular for model evaluation, but also to generate learning data. In this way, they can be reset and receive an action that modifies their current state, returning a new state, a reward.

5.1.2 Datasets

To generate training data for a given MDP, we start by learning optimal policies for solving that MDP. Given the number of states, the number of actions, transitions, reward and termination functions, it is possible to find optimal policies with algorithms such as Value Iteration [63]. Once the optimal policy is learned, the MDP can be solved with the policy. However, this only generates a single trajectory, with the learned policies being deterministic. We associate a score with each MDP, which is the sum of the rewards obtained by applying the optimal policy. This score will then serve as a comparison to evaluate the different models.

In order to generate different trajectories to get closer to the scenario of the datasets used in the paper, we add a factor $\epsilon \in [0, 1]$. When generating a trajectory, with probability ϵ, the chosen action is random, otherwise it is given by the optimal policy. We considered numbers of states among 10^2, 10^3, 10^4, 10^5 and 10^6, numbers of actions among 2, 3, 4, 5, 10, 20, 50 and 100 and reward numbers from 2, 3, 4, 5 and 10. For a number of rewards n, we vary the rewards between $-n + 2$ and 1 (in case of success). We use deterministic transition and reward functions based on states and actions. Transitions and rewards are chosen randomly for each state-action pair. A state is terminal with probability $\frac{|\mathcal{S}|}{10^9}$. We ensure that it has at least one terminal state and that the maximum reward is associated with transitions leading to terminal states. We choose a probability of $\epsilon = 0.5$ for the alternation between the random policy and the optimal policy. These configurations allow us to generate different datasets that we use for training models. The MDPs are then reused during the evaluation. The generated dataset can be accessed here: https://drive.google.com/drive/folders/1D5g_wI-X5vGaUMhMPknBzXT3cxWaiFs_?usp=share_link.

5.2 Evaluation metric

Each MDP is associated with the score obtained by the optimal policy. The minimum score for a trajectory of size T is $T \times r_{min}$ with r_{min}, the minimum reward. The models are therefore trained on the trajectories generated for the MDP and then evaluated based on their interactions with the environment. These interactions give rise to a trajectory, the score of which is calculated by adding the rewards obtained. The score is then normalized and reduced to a percentage based on the interval of the minimum possible score and the MDP reference score. In the paper, the reference score on Atari environments for example is given by the score obtained by an expert human player [12].

5.3 Experiments and results

5.3.1 Decision Transformer on different MDPs

Our first experiment consisted in training and then evaluating Decision Transformer on the various MDPs, whose generation was described in the previous section. For all MDPs, we keep the same Decision Transformer structure with the hyper-parameters listed in Table 1. We chose an Adam optimizer with a learning rate of $6e{-}4$ and a weight decay of 0 and betas = $(0.9, 0.95)$. The models are trained over 5 epochs. During evaluation, the maximum number of steps in the environment is 1000. We report the model score for the different configurations of number of states, number of actions and number of rewards in Table 2.

Hyper-parameter	N blocks	N heads	d_{model}	K
Value	6	8	128	12

Table 1: Decision Transformer hyper-parameters

The model's performance is good on average. For MDPs with 100 states, the model performs better than MDPs with more states. We suspect that this is due to the fact that the models are trained with the same number of epochs and evaluated on the same number of steps; whereas the more states and actions, the greater the complexity of the model. Comparing these results with those of the paper [12], where the Decision Transformer score reaches and exceeds 100%, the difference is that the reference score of our MDPs is the score of the optimal trajectory given by the optimal policy. Moreover, there is a lot of randomness (0.5 for the random policy) in the generation of our training data, which produces sub-optimal trajectories. We note that, despite this highly constraining configuration for learning on several points, the performance of the Decision Transformer model is very good. Thus, training the model on trajectory data unperturbed by randomness, adapting the number of epochs and the hyper-parameters of the model according to the complexity of the environment, would yield even more interesting results.

5.3.2 Decision Transformer structure

After freezing the structure and hyper-parameters of Decision Transformer to study its performance on different MDPs in the previous experiment, for this experiment we are studying the effects of changing the model structure on a single MDP. We have chosen an MDP with 1000 states, 10 actions and 5 possible rewards.

The different structures of the model were trained over 5 epochs and evaluated over 1000 steps maximum. We notice that on average the performance of the model becomes better with more blocks of transformers and more attention heads per block.

Some configurations of the model clearly show that the Decision Transformer model is capable of solving reinforcement learning problems, even in cases of disturbed data with random policies.

Actions	Rewards	States				
		10^2	10^3	10^4	10^5	10^6
2	3	93.71	42.84	34.37	50.55	99.70
	4	52.36	61.61	54.27	46.34	77.50
	5	33.43	34.21	53.92	52.51	51.99
	10	98.69	69.82	52.53	60.51	63.89
		69.55 ± 27.54	52.12 ± 14.23	48.77 ± 8.34	52.48 ± 5.15	73.27 ± 17.73
3	3	74.33	62.74	48.15	50.45	82.92
	4	65.68	50.03	58.85	49.10	51.28
	5	52.20	46.73	38.06	51.77	-
	10	99.37	74.40	55.86	50.01	-
		72.89 ± 17.20	58.48 ± 10.97	50.23 ± 8.04	50.33 ± 0.96	-
4	3	97.80	54.10	48.95	56.34	-
	4	42.85	50.10	49.87	62.27	-
	5	58.25	54.70	45.48	50.93	-
	10	99.69	49.74	59.20	47.69	-
		74.65 ± 24.71	52.16 ± 2.25	50.88 ± 5.08	54.31 ± 5.54	-
5	3	42.66	68.63	51.15	56.54	-
	4	55.12	54.32	78.41	51.32	-
	5	96.30	51.15	45.75	52.40	-
	10	69.22	45.21	59.74	50.58	-
		65.82 ± 19.95	54.83 ± 8.61	58.76 ± 12.39	52.71 ± 2.30	-
10	3	99.90	35.96	45.35	50.45	-
	4	97.95	12.64	48.68	51.92	-
	5	97.67	55.28	54.05	51.15	-
	10	94.86	52.31	59.20	49.74	-
		97.60 ± 1.80	39.05 ± 16.93	51.82 ± 5.27	50.82 ± 0.81	-
20	3	99.50	38.16	66.73	48.35	-
	4	89.06	49.18	52.17	58.72	-
	5	49.95	46.12	43.19	49.58	-
	10	99.94	36.50	49.47	55.52	-
		84.61 ± 20.48	42.49 ± 5.30	52.89 ± 8.63	53.04 ± 4.25	-
50	3	99.40	46.25	53.15	49.25	-
	4	99.60	78.41	49.88	50.92	-
	5	53.08	45.75	56.65	48.55	-
	10	32.33	59.74	49.76	48.16	-
		71.10 ± 29.33	57.54 ± 13.29	52.36 ± 2.83	49.22 ± 1.06	-
100	3	-	-	-	50.65	-
	4	-	-	-	91.40	-
	5	-	-	-	48.22	-
	10	-	-	-	47.14	-
		-	-	-	59.35 ± 18.55	-

Table 2: Decision Transformer scores on MDPs for different configurations of number of states, number of actions and number of rewards. We report for each state-action configuration, the mean and standard deviation of the reward configurations.

		Blocks					
h	d_{model}	**1**	**2**	**4**	**6**	**8**	
1	16	70.61	68.71	76.94	51.04	40.38	61.54 ± 13.63
	32	09.69	84.47	51.04	41.81	81.43	53.69 ± 27.57
	64	71.24	98.33	49.61	87.53	93.70	80.08 ± 17.78
	128	55.38	50.11	53.64	36.72	51.08	49.39 ± 06.60
	256	50.71	47.88	51.24	33.48	51.24	46.91 ± 06.83
		51.53 ± 22.44	69.90 ± 19.49	56.49 ± 10.30	50.12 ± 19.63	63.57 ± 20.36	
2	16	64.07	54.28	53.31	97.46	99.06	73.64 ± 20.46
	32	51.94	42.85	51.41	56.14	47.71	50.01 ± 04.47
	64	50.71	89.63	98.53	98.83	86.87	84.91 ± 17.75
	128	45.45	48.75	96.76	79.30	50.61	64.17 ± 20.31
	256	52.91	60.07	57.01	51.24	51.28	54.50 ± 03.49
		53.02 ± 06.10	59.12 ± 16.29	71.40 ± 21.51	76.59 ± 19.99	67.11 ± 21.50	
4	16	39.95	52.48	57.11	96.10	49.20	58.97 ± 19.40
	32	57.28	92.56	52.88	71.80	91.43	73.19 ± 16.59
	64	45.18	98.93	52.54	92.70	70.10	71.89 ± 21.24
	128	52.94	61.91	51.54	95.50	51.24	62.63 ± 16.90
	256	50.01	49.88	51.24	51.24	39.95	48.46 ± 04.30
		49.07 ± 06.03	71.15 ± 20.57	53.06 ± 02.11	81.47 ± 17.57	60.38 ± 18.35	
8	16	61.44	91.23	56.41	52.31	53.88	63.05 ± 14.42
	32	57.61	54.58	93.90	50.18	88.90	69.03 ± 18.48
	64	40.35	51.28	47.65	49.28	92.06	56.12 ± 18.34
	128	39.55	46.71	95.36	55.28	51.24	57.63 ± 19.58
	256	52.24	50.98	41.68	39.95	39.95	44.96 ± 05.48
		50.24 ± 08.90	58.96 ± 16.33	67.00 ± 23.05	49.40 ± 05.16	65.21 ± 21.18	

Table 3: Decision Transformer scores on MDPs for different configurations of number of blocks, number of heads and embedding dimension.

6 Conclusions

We worked on the paper Decision Transformer: Reinforcement Learning via Sequence Modeling, which presents an approach seeking to abstract reinforcement learning into a sequence modeling task while leveraging the effectiveness of the Transformer model in this task. For simplicity reasons, we decided to evaluate Decision Transformer on simple MDPs. To generate learning trajectories, we start by learning optimal policies to solve these MDPs. Then, the trajectories are generated by alternating between the optimal policy and a random policy, so that we can find ourselves in a more complex learning configuration. The various experiments that we have done as well as the experiments carried out by the authors of the paper show that Decision Transformer can be an architecture of choice for tackling offline reinforcement learning problems, compared to conventional architectures and modeling approaches.

As the training of the model is done on offline data, it is important to ensure the reliability of this data for real applications, because they condition the outputs just as in classic reinforcement learning. This work is part of a new approach to modeling tasks in reinforcement learning by taking advantage of the efficiency of transformers models. Current and future work further exploits how to effectively integrate transformers into reinforcement learning.

References

[1] ABRAMSON, J., AHUJA, A., BARR, I., BRUSSEE, A., CARNEVALE, F., CASSIN, M., CHHAPARIA, R., CLARK, S., DAMOC, B., DUDZIK, A., ET AL. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672* (2020).

[2] AGARWAL, R., SCHUURMANS, D., AND NOROUZI, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning* (2020).

[3] AJAY, A., KUMAR, A., AGRAWAL, P., LEVINE, S., AND NACHUM, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611* (2020).

[4] ARJONA-MEDINA, J. A., GILLHOFER, M., WIDRICH, M., UNTERTHINER, T., BRANDSTETTER, J., AND HOCHREITER, S. Rudder: Return decomposition for delayed rewards. *arXiv preprint arXiv:1806.07857* (2018).

[5] BA, J. L., KIROS, J. R., AND HINTON, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] BELLEMARE, M. G., NADDAF, Y., VENESS, J., AND BOWLING, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research 47* (2013), 253–279.

[7] BELLMAN, R. A markovian decision process. *Journal of Mathematics and Mechanics 6*, 5 (1957), 679–684. JSTOR 24900506.

[8] BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[9] BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., AND ASKELL, AMANDA, E. A. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[10] CAMPOS, V., TROTT, A., XIONG, C., SOCHER, R., GIRO-I NIETO, X., AND TORRES, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning* (2020).

[11] CARION, N., MASSA, F., SYNNAEVE, G., USUNIER, N., KIRILLOV, A., AND ZAGORUYKO, S. End-to-end object detection with transformers. In *European Conference on Computer Vision* (2020).

[12] CHEN, L., LU, K., RAJESWARAN, A., LEE, K., GROVER, A., LASKIN, M., ABBEEL, P., SRINIVAS, A., AND MORDATCH, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345* (2021).

[13] CHEN, M., RADFORD, A., CHILD, R., WU, J., JUN, H., LUAN, D., AND SUTSKEVER, I. Generative pretraining from pixels. In *International Conference on Machine Learning* (2020), PMLR, pp. 1691–1703.

[14] DASARI, S., AND GUPTA, A. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970* (2020).

[15] DATHATHRI, S., MADOTTO, A., LAN, J., HUNG, J., FRANK, E., MOLINO, P., YOSINSKI, J., AND LIU, R. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164* (2019).

[16] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[17] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., ET AL. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[18] ECOFFET, A., HUIZINGA, J., LEHMAN, J., STANLEY, K. O., AND CLUNE, J. Go-explore: A new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995* (2019).

[19] EYSENBACH, B., GUPTA, A., IBARZ, J., AND LEVINE, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations* (2019).

[20] FERRET, J., MARINIER, R., GEIST, M., AND PIETQUIN, O. Self-attentional credit assignment for transfer in reinforcement learning. *arXiv preprint arXiv:1907.08027* (2019).

[21] FICLER, J., AND GOLDBERG, Y. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633* (2017).

[22] FU, J., KUMAR, A., NACHUM, O., TUCKER, G., AND LEVINE, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[23] FUJIMOTO, S., MEGER, D., AND PRECUP, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning* (2019).

[24] GAO, Y., XU, H., LIN, J., YU, F., LEVINE, S., AND DARRELL, T. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313* (2018).

[25] GHAZVININEJAD, M., SHI, X., PRIYADARSHI, J., AND KNIGHT, K. Hafez: An interactive poetry generation system. In *Proceedings of ACL, System Demonstrations* (2017).

[26] GHOSH, D., GUPTA, A., FU, J., REDDY, A., DEVIN, C., EYSENBACH, B., AND LEVINE, S. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088* (2019).

[27] HAFNER, D., LILLICRAP, T., NOROUZI, M., AND BA, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193* (2020).

[28] HARUTYUNYAN, A., DABNEY, W., MESNARD, T., AZAR, M., PIOT, B., HEESS, N., VAN HASSELT, H., WAYNE, G., SINGH, S., AND PRECUP, DOINA, E. A. Hindsight credit assignment. *arXiv preprint arXiv:1912.02503* (2019).

[29] HOLTZMAN, A., BUYS, J., FORBES, M., BOSSELUT, A., GOLUB, D., AND CHOI, Y. Learning to write with cooperative discriminators. *arXiv preprint arXiv:1805.06087* (2018).

[30] HU, Z., YANG, Z., LIANG, X., SALAKHUTDINOV, R., AND XING, E. P. Toward controlled generation of text. In *International Conference on Machine Learning* (2017).

[31] HUNG, C.-C., LILLICRAP, T., ABRAMSON, J., WU, Y., MIRZA, M., CARNEVALE, F., AHUJA, A., AND WAYNE, G. Optimizing agent behavior over long time scales by transporting value. *Nature Communications 10*, 1 (2019), 1–12.

[32] JANNER, M., FU, J., ZHANG, M., AND LEVINE, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems* (2019), pp. 12498–12509.

[33] JANNER, M., LI, Q., AND LEVINE, S. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039* (2021).

[34] KARRAS, T., LAINE, S., AND AILA, T. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition* (2019).

[35] KESKAR, N. S., MCCANN, B., VARSHNEY, L. R., XIONG, C., AND SOCHER, R. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* (2019).

[36] KIDAMBI, R., RAJESWARAN, A., NETRAPALLI, P., AND JOACHIMS, T. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems* (2020).

[37] KRAUSE, B., GOTMARE, A. D., MCCANN, B., KESKAR, N. S., JOTY, S., SOCHER, R., AND RAJANI, N. F. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367* (2020).

[38] KUMAR, A., FU, J., TUCKER, G., AND LEVINE, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949* (2019).

[39] KUMAR, A., PENG, X. B., AND LEVINE, S. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465* (2019).

[40] KUMAR, A., ZHOU, A., TUCKER, G., AND LEVINE, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems* (2020).

[41] LEVINE, S., KUMAR, A., TUCKER, G., AND FU, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[42] LIU, Y., LUO, Y., ZHONG, Y., CHEN, X., LIU, Q., AND PENG, J. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420* (2019).

[43] LOSHCHILOV, I., AND HUTTER, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[44] LU, K., GROVER, A., ABBEEL, P., AND MORDATCH, I. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548* (2020).

[45] MESNARD, T., WEBER, T., VIOLA, F., THAKOOR, S., SAADE, A., HARUTYUNYAN, A., DABNEY, W., STEPLETON, T., HEESS, N., AND GUEZ, ARTHUR, E. A. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464* (2020).

[46] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013). NIPS Deep Learning Workshop 2013.

[47] MNIH, V., KAVUKCUOGLU, K., SILVER, D., RUSU, A. A., VENESS, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M., FIDJELAND, A. K., AND OSTROVSKI, GEORG, E. A. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529–533.

[48] NAIR, A., DALAL, M., GUPTA, A., AND LEVINE, S. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[49] PARISOTTO, E., SONG, F., RAE, J., PASCANU, R., GULCEHRE, C., JAYAKUMAR, S., JADERBERG, M., LOPEZ KAUFMAN, R., CLARK, A., AND NOURY, SEB, E. A. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning* (2020).

[50] PASTER, K., MCILRAITH, S. A., AND BA, J. Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419* (2020).

[51] PENG, X. B., KUMAR, A., ZHANG, G., AND LEVINE, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177* (2019).

[52] PERTSCH, K., LEE, Y., AND LIM, J. J. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944* (2020).

[53] RADFORD, A., NARASIMHAN, K., SALIMANS, T., AND SUTSKEVER, I. Improving language understanding by generative pre-training.

[54] RAJANI, N. F., MCCANN, B., XIONG, C., AND SOCHER, R. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361* (2019).

[55] RAMESH, A., PAVLOV, M., GOH, G., GRAY, S., VOSS, C., RADFORD, A., CHEN, M., AND SUTSKEVER, I. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092* (2021).

[56] RAPOSO, D., RITTER, S., SANTORO, A., WAYNE, G., WEBER, T., BOTVINICK, M., VAN HASSELT, H., AND SONG, F. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425* (2021).

[57] RITTER, S., FAULKNER, R., SARTRAN, L., SANTORO, A., BOTVINICK, M., AND RAPOSO, D. Rapid task-solving in novel environments. *arXiv preprint arXiv:2006.03662* (2020).

[58] SHARMA, A., GU, S., LEVINE, S., KUMAR, V., AND HAUSMAN, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations* (2020).

[59] SIEGEL, N. Y., SPRINGENBERG, J. T., BERKENKAMP, F., ABDOLMALEKI, A., NEUNERT, M., LAMPE, T., HAFNER, R., AND RIEDMILLER, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. In *International Conference on Learning Representations* (2020).

[60] SINGH, A., LIU, H., ZHOU, G., YU, A., RHINEHART, N., AND LEVINE, S. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations* (2021).

[61] SRIVASTAVA, R. K., SHYAM, P., MUTZ, F., JASKOWSKI, W., AND SCHMIDHUBER, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877* (2019).

[62] SUTTON, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML* (1990).

[63] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[64] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017).

[65] WATKINS, C., AND DAYAN, P. Q-learning. *Machine Learning 8* (1992), 279–292.

[66] WENG, L. Controllable neural text generation, 2021.

[67] WOLF, T., CHAUMOND, J., DEBUT, L., SANH, V., DELANGUE, C., MOI, A., CISTAC, P., FUNTOWICZ, M., DAVISON, J., SHLEIFER, S., ET AL. Transformers: State-of-the-art natural language processing. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (2020).

[68] WU, Y., TUCKER, G., AND NACHUM, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[69] YU, L., ZHANG, W., WANG, J., AND YU, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence* (2017).

[70] YU, T., THOMAS, G., YU, L., ERMON, S., ZOU, J., LEVINE, S., FINN, C., AND MA, T. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems* (2020).

[71] ZAMBALDI, V., RAPOSO, D., SANTORO, A., BAPST, V., LI, Y., BABUSCHKIN, I., TUYLS, K., REICHERT, D., LILLICRAP, T., AND LOCKHART, EDWARD, E. A. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations* (2018).

[72] ZIEGLER, D. M., STIENNON, N., WU, J., BROWN, T. B., RADFORD, A., AMODEI, D., CHRISTIANO, P., AND IRVING, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).