

Return-Aligned Decision Transformer

Tsunehiko Tanaka^{1,2} Kenshi Abe² Kaito Ariu² Tetsuro Morimura² Edgar Simo-Serra¹

Abstract

Traditional approaches in offline reinforcement learning aim to learn the optimal policy that maximizes the cumulative reward, also known as return. However, as applications broaden, it becomes increasingly crucial to train agents that not only maximize the returns, but align the actual return with a specified target return, giving control over the agent’s performance. Decision Transformer (DT) optimizes a policy that generates actions conditioned on the target return through supervised learning and is equipped with a mechanism to control the agent using the target return. Despite being designed to align the actual return with the target return, we have empirically identified a discrepancy between the actual return and the target return in DT. In this paper, we propose Return-Aligned Decision Transformer (RADT), designed to effectively align the actual return with the target return. Our model decouples returns from the conventional input sequence, which typically consists of returns, states, and actions, to enhance the relationships between returns and states, as well as returns and actions. Extensive experiments show that RADT reduces the discrepancies between the actual return and the target return of DT-based methods.

1. Introduction

Offline reinforcement learning (RL) focuses on learning optimal policies using trajectories from offline datasets (Levine et al., 2020; Fujimoto & Gu, 2021; Jin et al., 2021; Xu et al., 2022). Many methods in offline RL traditionally aim to learn the optimal policy that maximizes the cumulative reward, also known as return. However, in various scenarios, it is crucial to train an agent to align the cumulative reward, which we refer to as the *actual return*, with a given target value, which we call *target return*. Through specifying the

¹Waseda University, Tokyo, Japan ²CyberAgent, Tokyo, Japan. Correspondence to: Tsunehiko Tanaka <tsunehiko@fuji.waseda.jp>.

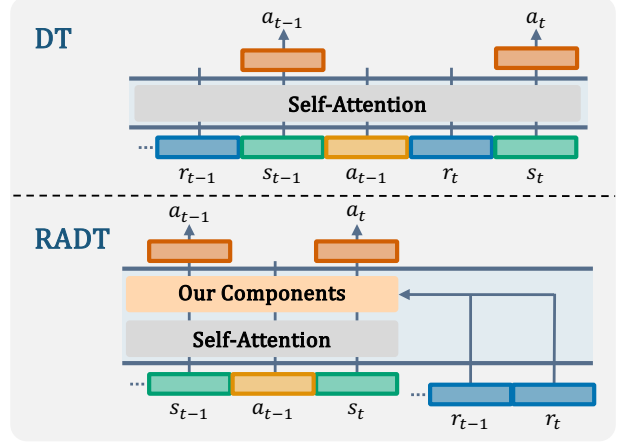


Figure 1. Comparison between DT and the proposed RADT architecture. RADT decouples the returns from states and actions in the input sequence, explicitly modeling the relationships between returns and other modalities.

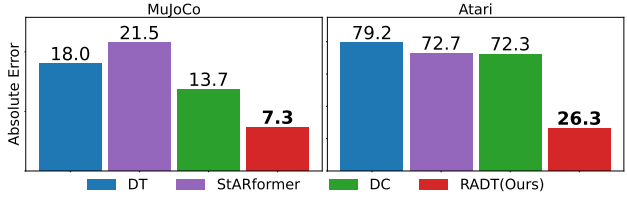


Figure 2. Absolute errors between the target return and the actual return in both the MuJoCo and Atari domains. The discrepancies are normalized by the return range between the top 5% and bottom 95% within the dataset. RADT significantly reduces the discrepancies in both domains.

actual return, it is possible to adjust the performance of game bots (Shen et al., 2021) or educational tools (Singla et al., 2021) to match user skill levels, or model the trajectories of non-expert agents in traffic simulation analysis (Nguyen et al., 2021). Despite this importance, existing approaches exhibit discrepancies between the actual return and target return, which significantly lower the quality of the agents. In this work, we aim to align the actual return with the target return, enabling control of the agent’s performance based on the target return.

Decision Transformer (DT) (Chen et al., 2021) optimizes

a policy that generates actions conditioned on the target return through supervised learning, and is equipped with a mechanism to control the agent using the target return. Specifically, this model takes a sequence comprising future desired returns, past states, and actions as inputs, and outputs actions using a transformer architecture (Vaswani et al., 2017). The self-attention mechanism in the transformer extracts significant features of the input sequence based on the relative importance of each token to all the other tokens. DT integrates returns into the input sequence to condition action generation. However, as illustrated in Fig. 2, there are discrepancies between the actual returns and the target returns. This error could be explained by the simultaneous handling of returns with the other modalities such as states and actions. This simultaneous processing could lead the model to overly focus on state or action tokens, and it may become unable to align the actual return with the target return appropriately.

In this paper, we propose a novel architecture, *Return-Aligned Decision Transformer* (RADT), designed to align the actual return with the target return, as shown in Fig. 1. The core idea is to use an architecture that separates returns from the input sequence and explicitly models the relationships between the returns and the other modalities. This architecture prevents the reduction of the target return’s influence on the output actions due to other modalities such as state and action. To achieve this, we employ two techniques. The first is a unique cross-attention mechanism that focuses on the relationship between a state-action sequence and a return sequence. The second is adaptive layer normalization, which scales the state-action features using parameters inferred solely from the return features. They are specifically designed for seamless integration into the transformer block, thus maintaining the simple supervised learning approach of DT. In our experiments, RADT shows a significant reduction in the absolute error between the actual return and the target return, decreasing it to 39.7% of DT’s error in the MuJoCo (Todorov et al., 2012) domain and 29.8% in the Atari (Bellemare et al., 2013) domain. We evaluate the key techniques of RADT through an ablation study, demonstrating that each technique is effective for different tasks and can complement each other.

Our contributions are summarized as follows:

- We propose a novel approach for DT designed to align the actual return with a given target return.
- To realize this concept, we present Return-Aligned Decision Transformer, which decouples the returns from states and actions in the input sequence, explicitly modeling the relationships between returns and other modalities.
- We introduce a unique cross-attention mechanism and

adaptive layer normalization to model the relationships between returns and other modalities.

- Our experiments demonstrate that our method outperforms existing approaches in aligning actual returns with target returns.

2. Preliminary

We assume a finite horizon Markov Decision Process (MDP) with horizon T as our environment, which can be described as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mu, P, R \rangle$, where $\mathcal{S}$ represents the state space; $\mathcal{A}$ represents the action space; $\mu \in \Delta(\mathcal{S})$ represents the initial state distribution; $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ represents the transition probability distribution; and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward function. The environment begins from an initial state s_1 sampled from a fixed distribution μ . At each timestep $t \in [T]$, an agent takes an action $a_t \in \mathcal{A}$ in response to the state $s_t \in \mathcal{S}$, transitioning to the next state $s_{t+1} \in \mathcal{S}$ with the probability distribution $P(\cdot | s_t, a_t)$. Concurrently, the agent receives a reward $r_t = R(s_t, a_t)$.

Decision Transformer (DT) introduces the paradigm of transformers into the context of offline reinforcement learning. At each timestep t in the inference, DT takes a sequence of desired returns, past states, and actions as inputs, and outputs an action a_t . The input sequence of DT is represented as

$$\tau = (\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \dots, \hat{R}_t, s_t), \quad (1)$$

where $\hat{R}_t$ represents the returns computed over the remaining steps¹. It is calculated as $\hat{R}_t = R^{\text{target}} - \sum_{t'=1}^{t-1} r_{t'}$. R^{target} is a given constant², which is the total desired return to be obtained in an episode of length T . We refer to R^{target} as a target return. Raw inputs, referred to as tokens, are individually projected into the embedding dimension by separate learnable linear layers for return, state, and action respectively, to generate token embeddings. The tokens are processed using a Transformer-based GPT model (Radford et al., 2018). The model is trained using either cross-entropy or mean-squared error loss, calculated between the predicted action and the ground truth from the offline datasets.

The transformer is an architecture designed for processing sequential data, including the attention mechanism, residual connection, and layer normalization. The attention mechanism processes three distinct inputs: the query, the key, and the value. This process involves weighting the value by the normalized dot product of the query and the key. The i -th output token of the attention mechanism is calculated

¹Despite Eq. (1) taking inputs from 1 to t , in practice, only the last K timesteps are processed.

² R^{target} is the total return of the trajectory in the dataset during training, and it is a given constant during inference.

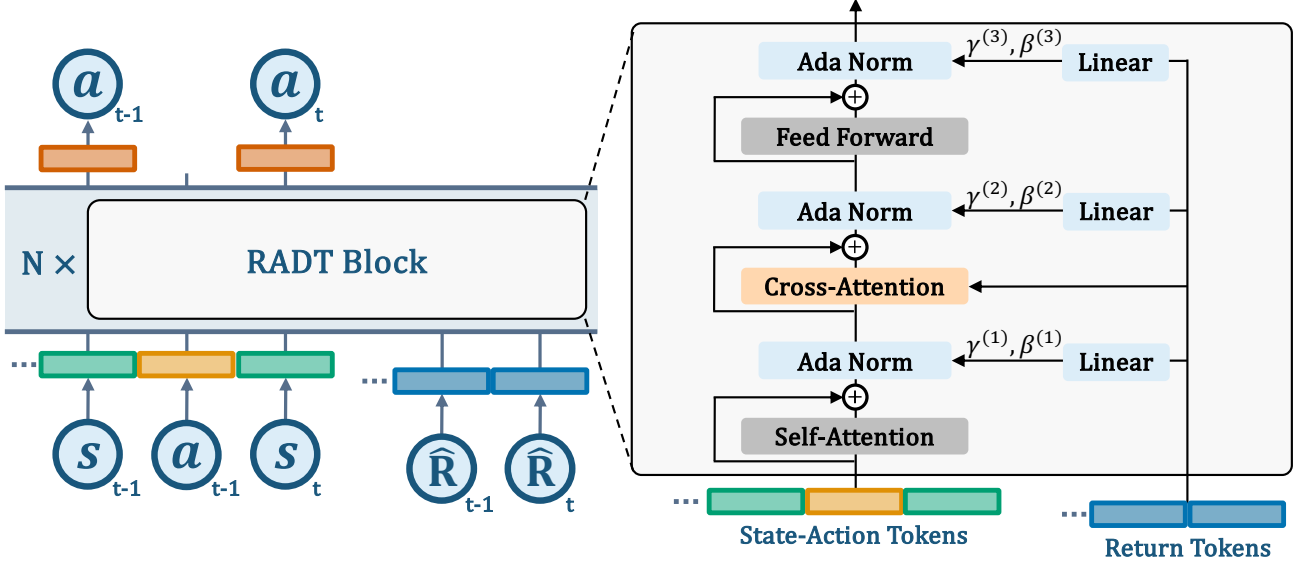


Figure 3. An overview of our proposed Return-Aligned Decision Transformer (RADT). We split input tokens into the return sequence and state-action sequence. RADT equips the transformer block with cross-attention and adaptive layer normalization to highlight returns tokens.

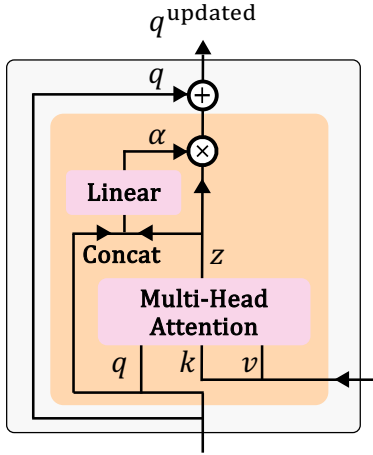


Figure 4. The detailed structure of our cross-attention. In the cross-attention, the state-action sequence is input as the query, and the return sequence serves as both key and value. The query input q and output z of the multi-head attention are fed into a linear layer to calculate the weight α for the output z .

as follows:

$$z_i = \sum_{j=1}^n \text{softmax}(\langle q_i, k_\ell \rangle_{\ell=1}^n)_j \cdot v_j, \quad (2)$$

DT uses self-attention, where query, key, and value are obtained by linearly different transformations of the input

sequence,

$$q_i = \tau_i W_i^Q, \quad k_i = \tau_i W_i^K, \quad v_i = \tau_i W_i^V, \quad (3)$$

Moreover, it employs a causal mask to prohibit attention to subsequent tokens, rendering tokens in future timesteps ineffective for action prediction.

Layer normalization standardizes the token features to stabilize learning. Residual connection avoids gradient vanishing by adding the input and output of attention layers or feed-forward layers. For further details on DT, refer to the original paper (Chen et al., 2021).

Our goal is to minimize the following absolute error between the target return and the actual return for any given target return R^{target} with a single model:

$$\mathbb{E} \left[\left| R^{\text{target}} - \sum_{t=1}^T r_t \right| \right]. \quad (4)$$

We refer to $\sum_{t=1}^T r_t$ as an actual return.

3. Return-Aligned Decision Transformer

In this section, we present the Return-Aligned Decision Transformer (RADT), a model designed to align the actual return with the target return. While DT constructs the input sequence using returns, states, and actions, we propose to separate returns from the input sequence. RADT incorporates techniques to explicitly model the relationships

between the returns and the other modalities, states and actions. The training process follows a supervised learning paradigm of DT. We show the overview of RADT in Fig. 3.

3.1. Overview

We separately input the return sequence ϕ and the state-action sequence τ^- into our model.

$$\phi = (\hat{R}_1, \hat{R}_2, \dots, \hat{R}_T), \quad (5)$$

$$\tau^- = (s_1, a_1, s_2, a_2, \dots, s_T). \quad (6)$$

DT integrates returns, states, and actions into a single input sequence to generate actions based on returns. However, this approach could diminish the attention allocated to the return tokens, leading to discrepancies between the actual return and the target return. This is because the self-attention mechanism excessively focuses on modalities other than returns. To address this, we split input tokens into the return and state-action sequences and apply an architecture that focuses more on the returns. The details of our architecture will be explained in the following section.

3.2. Architecture

We propose two techniques for RADT to explicitly model the relationships between the returns and the other modalities: cross-attention and adaptive layer normalization. Both techniques can be seamlessly integrated into transformer blocks, and we find that employing them in tandem yields superior performance. The architecture utilizing both techniques is illustrated on the right side of Fig. 3.

Cross-Attention between Returns and States-Actions. We apply cross-attention between the state-action sequence τ^- and the return sequence ϕ . The detailed structure of our cross-attention is shown in Fig. 4. In our cross-attention, the state-action sequence τ^- acts as the query, while the return sequence ϕ acts as both the key and value,

$$q_i = \tau_i^- W_i^Q, \quad k_i = \phi_i W_i^K, \quad v_i = \phi_i W_i^V. \quad (7)$$

These query, key, and value are input into multi-head attention (Eq. 2), and we obtain the output z . Note that we use a causal mask to ensure that tokens in the state-action sequence τ^- cannot access future return tokens.

Following the powerful cross-attention-based technique for integrating two different types of features (Nguyen et al., 2022), we adaptively adjust the scale for the output against the input of the multi-head attention. We concatenate the multi-head attention input q and output z as a column vector $[z_i; q_i] \in \mathbb{R}^{2D}$ and obtain dimension-wise scaling parameters α through a learnable affine projection. Using the scale

α , we define residual connection as

$$\alpha_i = W[z_i; q_i] + b, \quad (8)$$

$$q_i^{\text{updated}} = (1 + \alpha_i) \otimes z_i + q_i, \quad (9)$$

where $W \in \mathbb{R}^{D \times 2D}$ and $b \in \mathbb{R}^D$ are learnable parameters, and $\otimes$ denotes the Hadamard product. The choice of $1 + \alpha_i$ allows the model to start with a baseline scaling of 1 (simple addition) by zero-initialization of W and b , resulting in the scale of z_i and q_i being the same. This provides a stable starting point, from which the model can learn to adaptively adjust the scaling. As the training progresses, α_i is updated to refine the balance between z_i and q_i .

In the self-attention in DT, the input sequence τ , which is a concatenation of returns, states, and actions, is input as the query, key, and value. This setup attempts to express all interrelations solely through W^Q, W^K, W^V . However, the model gets distracted from such complicated relationships, making it less responsive to the returns. We address this by using cross-attention to focus on the relationships between returns and the other modalities. Additionally, by introducing a scale α for the output z relative to the input query q , we can more flexibly represent the relationship between the returns sequence ϕ and the state-action sequence τ^- .

Adaptive Layer Normalization. We next introduce adaptive layer normalization, which adaptively changes depending on the returns. This involves replacing the standard layer normalization layers in transformer blocks with adaptive layer normalization layers.

$$q'_i = (1 + \gamma_i) \otimes \text{LayerNorm}(q_i) + \beta_i \quad (10)$$

where q_i, q'_i denote the input and output of adaptive layer normalization, respectively. γ, β are dimension-wise scaling parameters regressed by a linear layer from the return sequence ϕ . Through this regression, the model can condition the state-action sequence τ^- with the return. Similar to the $1 + \alpha_i$ in Eq. (8), by zero-initializing the parameters of the linear layer, Eq. (10) can be considered the same as the standard layer normalization at the beginning of training.

In DT, layer normalization is positioned before either the attention layer or the feed-forward layer. This configuration creates pathways that bypass layer normalization via a residual connection. We place adaptive layer normalization after residual connection to ensure that all pathways undergo adaptive layer normalization, as in the vanilla transformer (Vaswani et al., 2017).

4. Experiments

In this section, we conduct extensive experiments to evaluate the performance of our RADT. First, we verify that RADT is effective in earning returns consistent with various

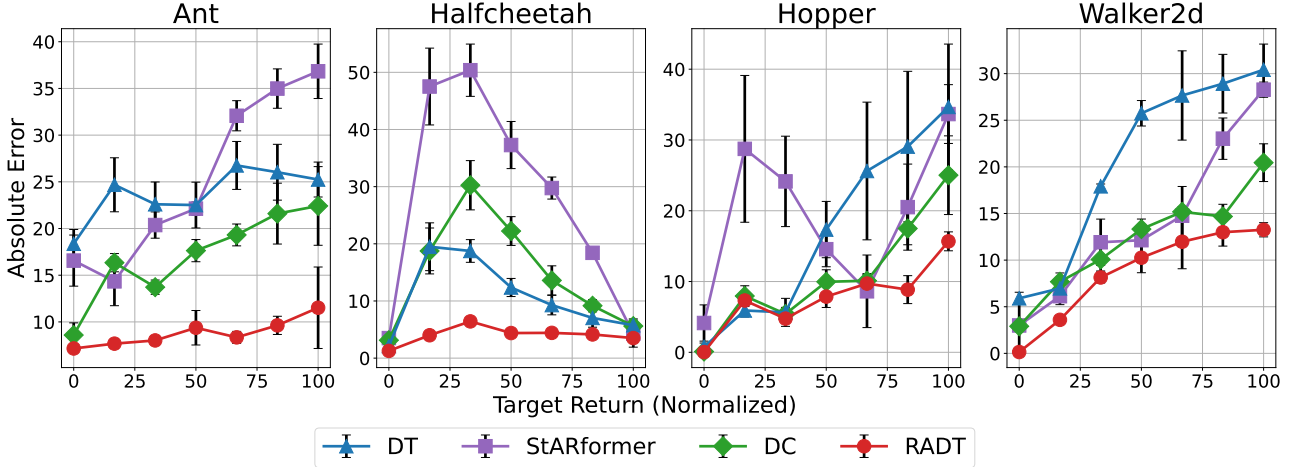


Figure 5. **Absolute Error Comparison of RADT and baselines in MuJoCo domain.** Each column represents a task. The x-axis is the target return. Target returns are divided into seven equally spaced points based on the cumulative reward of trajectories in the dataset, ranging from the bottom 5% to the top 5%. In this graph, the bottom 5% is represented as 0, and the top 5% as 100. The y-axis represents the absolute error between the actual return and the target return obtained from our simulations. We report the mean and standard error over three seeds.

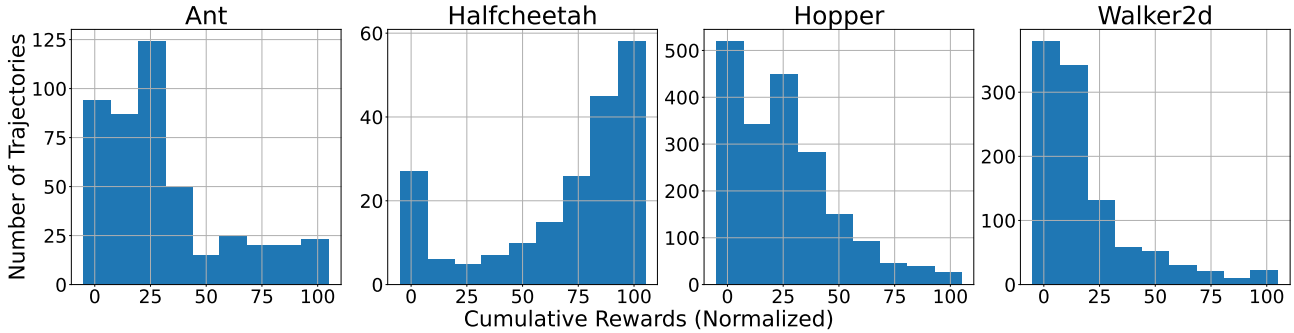


Figure 6. **Histogram of the cumulative reward of trajectories included in the D4RL dataset.** Each column represents a MuJoCo task. The x-axis is normalized by the top 5% and bottom 5% of the dataset’s trajectory cumulative reward, similar to Fig. 5. When viewed alongside Fig. 5, we observe that the absolute errors of DT and DC are inversely proportional to the amount of data, RADT consistently maintains a low absolute error regardless of the amount of data.

given target returns, compared to other baselines. Next, we demonstrate through an ablation study that the two types of architectural designs constituting RADT are effective individually, and that using both types together further improves performance. Additionally, we show that during training, RADT converges to the given target promptly.

4.1. Datasets

We evaluate RADT on continuous (MuJoCo (Todorov et al., 2012)) and discrete (Atari (Bellemare et al., 2013)) control tasks in the same way as DT. MuJoCo is characterized by a continuous action space with dense rewards. We use four gym locomotion tasks from the widely-used D4RL (Fu et al., 2020) dataset: Ant, Hopper, HalfCheetah, and Walker. In this experiment, we utilize the v2 medium-replay dataset.

The medium-replay comprises a replay buffer of policies that achieve approximately 1/3 of expert performance. Atari involves high-dimensional visual observations and requires capturing long-term context to handle the delay between actions and the resulting rewards. We use four tasks: Breakout, Pong, Qbert, and Seaquest. Similar to DT, we use 1% of all samples in the DQN-replay datasets as per Agarwal et al. (2020) for training.

4.2. Baselines and Settings

To evaluate our proposed model architecture, we use return-conditioned DT models with different architectures, specifically DT (Chen et al., 2021), StARformer (Shang et al., 2022), and Decision ConvFormer (DC) (Kim et al., 2024), as baselines. We use the official PyTorch implementations for

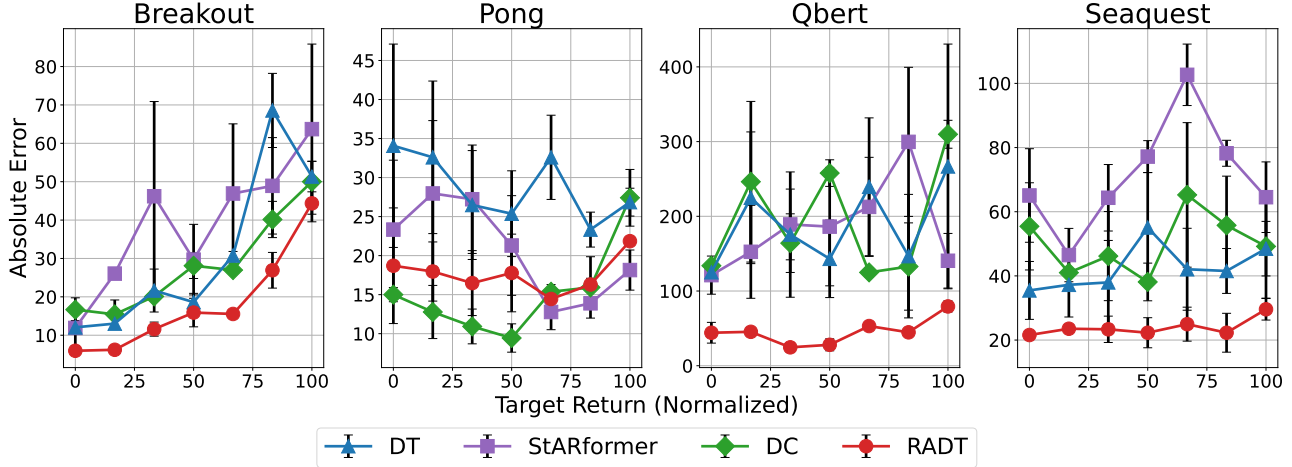


Figure 7. **Absolute Error Comparison of RADT and baselines in the Atari domain.** The way to read these graphs is the same as in Fig. 5. We report the mean and standard error over three seeds.

Table 1. **Performance comparison in the actual return maximization in the MuJoCo domain.** We cite the results for DT, DC, and ODT from their reported scores. The results for CQL are cited from [Kostrikov et al. \(2022\)](#). We report the average across 3 seeds from our simulation results.

Method	CQL	DT	DC	ODT	RADT
halfcheetah-m	44.0	42.6	43.0	42.2	43.1
hopper-m	58.5	67.6	92.5	97.5	71.5
walker2d-m	72.5	74.0	79.2	76.8	78.3
halfcheetah-m-r	45.5	36.6	41.3	40.4	41.6
hopper-m-r	95.0	82.7	94.2	88.9	95.1
walker2d-m-r	77.2	66.6	76.6	76.9	75.9
halfcheetah-m-e	91.6	86.8	93.0	-	93.4
hopper-m-e	105.4	107.6	110.4	-	111.7
walker2d-m-e	108.8	108.1	109.6	-	108.9

baselines. Although StARformer uses step-by-step rewards instead of returns, in our experiments, we employ return-conditioning using returns. This modification is stated in the original paper ([Shang et al., 2022](#)) to have minimal impact on performance. The hyperparameters for each method are set according to the defaults specified in their respective papers or open-source codebases. For visual observations in the Atari domain, RADT and DC use the same CNN encoder as DT. StARformer, in addition to the CNN encoder, also incorporates more powerful patch-based embeddings like Vision Transformer ([Dosovitskiy et al., 2021](#)).

In MuJoCo, for each method, we train three instances with different seeds, and each instance runs 20 episodes for each target return. In Atari, for each method, we train three instances with different seeds, and each instance runs 10 episodes for each target return. Target returns are set by first identifying the range of cumulative reward in trajectories

in the training dataset, specifically from the bottom 5% to the top 5%. This identified range is then equally divided into seven intervals, not based on percentiles, but by simply dividing the range into seven equal parts. Each of these parts represents a target return.

4.3. Main Results

The main results are presented in Fig. 5 for the MuJoCo domain and Fig. 7 for the Atari domain. These figures plot the absolute error between the actual return and the target return, where lower is better. Additionally, the target returns are represented with the top 5% values as 100 and the bottom 5% values as 0.

MuJoCo Domain. In the MuJoCo domain, as shown in Fig. 5, RADT outperforms the baseline in all tasks. In HalfCheetah, the absolute error of DT becomes smaller as the target return gets closer to 0 or 100, while in other tasks, the error decreases as the target return gets closer to 0. Additionally, we observe that DC exhibits a similar trend in absolute error to DT across all four tasks. These trends in absolute error are inversely proportional to the cumulative reward of trajectories in the dataset, as shown in Fig. 6. We believe that DT and DC are susceptible to the distribution of data. In contrast, RADT consistently shows smaller absolute errors across various target returns compared to DT and DC, indicating its robustness to data distribution.

It is noteworthy that in StARformer, the absolute error decreases as it gets close to 66.7 in Hopper and 100 in HalfCheetah. This is because, regardless of the specified target return, the actual return tends to converge to 66.7 (Hopper) or 100 (HalfCheetah), as shown in Fig. 9a in Appendix A.1. [Shang et al. \(2022\)](#) report that it performs

Table 2. Results of the ablation study on the proposed techniques proposed in Sec.3.3. Each value indicates the sum of absolute error between the actual return and the target return, relative to the target returns in the main results. We report the mean and standard error over 3 seeds, and normalize such that the average and variance of the results when using both techniques (RADT) are 1.0 respectively. CA represents cross-attention, and AdaLN stands for adaptive layer normalization.

Approach	CA	AdaLN	Ant	HalfCheetah	Hopper	Walker
DT			2.69±2.07	2.66±2.07	2.19±6.49	2.39±2.60
RADT w/o AdaLN	✓		1.29±0.38	1.51±1.19	1.20±1.55	1.24±2.66
RADT w/o CA		✓	1.25±0.33	1.17±1.77	1.61±1.72	1.26±2.48
RADT Full	✓	✓	1.00±1.00	1.00±1.00	1.00±1.00	1.00±1.00



Figure 8. Transitions of the absolute errors between the actual and target returns during training in HalfCheetah, comparing the absolute errors of DT, StARformer, and DC. The mean and standard error are reported across 3 seeds.

equally well with or without using returns or step-by-step rewards. We believe that StARformer generates actions independently of returns or step-by-step rewards, suggesting that it is challenging to control it with various target returns. Conversely, RADT does not exhibit the bias seen in StARformer and responds more accurately to the target returns.

Atari Domain. The Atari domain is challenging due to the sparsity of rewards, making the control of obtained rewards more difficult than in MuJoCo. Despite this, RADT outperforms the baselines in most tasks, as shown in Fig. 7. Moreover, given that observations in the Atari domain are images, enhancements to the vision encoder offer considerable advantages. However, when compared to StARformer, which utilizes powerful patch-based embeddings such as ViT, RADT shows superior performance in most tasks and achieves competitive results in others. DC reports excellent results for Pong, demonstrating that convolution is superior to self-attention in limited tasks. Meanwhile, our proposed method performs best in the other tasks, suggesting that RADT would be better suited for a wide range of scenarios.

4.4. Ability to Maximize Expected Return

We conduct experiments on the MuJoCo domain to examine the ability to maximize the expected return, a standard task in offline RL. We consider four baselines: CQL (Kumar et al., 2020), DT (Chen et al., 2021), DC (Kim et al., 2024), and ODT (Zheng et al., 2022). The results are shown in Tab. 1. All scores are normalized so that 100 represents the score of an expert policy, as per Fu et al. (2020). We can observe that RADT achieves competitive or superior results compared to the baselines. These results mean that the architectural design of RADT can not only align the actual return with the target return, but also excel in return maximization.

4.5. Ablation Study

Architecture Techniques. We conduct an ablation study for the core techniques of RADT, as proposed in Sec. 3.2, on the MuJoCo domain. The results are summarized in Tab. 2. CA represents cross-attention, and AdaLN stands for adaptive layer normalization. Each value is the sum of the absolute errors between the actual return and the target return across the multiple target returns. These values are then normalized

so that the average and standard error of the results when using both techniques (RADT) are 1.0 respectively. Table 2 reports that introducing either cross-attention or adaptive layer normalization individually proves to be effective. The cross-attention performs better in Hopper and Walker, while adaptive layer normalization excels in Ant and HalfCheetah. This suggests that each technique has its unique areas of expertise. The combination of both techniques results in the smallest error across all tasks, implying that cross-attention and adaptive layer normalization effectively complement each other.

Training Convergence. We present in Fig. 8 the curves of the absolute errors between the actual return and the target return as training proceeds, along with the absolute errors of DT, StARformer, and DC as baselines. For all target returns, we observe that RADT converges up to 20k iterations. This suggests that RADT has an advantage in terms of the speed and stability of training compared to other baselines.

5. Related Work

5.1. RTG-Conditioned Offline RL

Recent studies have focused on formulating offline reinforcement learning (RL) as a problem of predicting action sequences that are conditioned by goals and rewards (Chen et al., 2021; Emmons et al., 2022; David et al., 2023; Schmidhuber, 2019; Srivastava et al., 2019). This approach differs from the popular value-based methods (Kumar et al., 2020; Kostrikov et al., 2022) by modeling the relationship between rewards and actions through supervised learning. Decision Transformer (DT) (Chen et al., 2021) introduces the concept of desired future returns and improves performance by training the transformer (Vaswani et al., 2017) as a return-conditioned policy. There have been various advancements in offline RL based on the DT framework, which will be elaborated in the following section.

5.2. Transformers for RL

Some efforts focus on refining the transformer architecture for offline RL. StARformer (Shang et al., 2022) introduces two transformer architectures, one aggregates information at each step, and the other aggregates information across the entire sequence. The image encoding process is improved by dividing the observation images into patches and feeding them into the transformer to enhance step information, similar to Vision Transformer (Dosovitskiy et al., 2021). Decision ConvFormer (Kim et al., 2024) replaces attention with convolution to capture the inherent local dependence pattern of MDP. While these architectures preserve the input sequence structure of the transformer, comprising returns, states, and actions, they do not directly tackle the challenge of diminishing the influence of returns on the

decision-making process. In contrast, our research specifically aims to align the actual return with the target return.

There is growing interest in methods of manipulating data to train DT to solve the trajectory stitching problem in offline RL. Stitching refers to the ability to combine sub-optimal trajectories to create an optimal one. For example, Q-learning Decision Transformer (QDT) (Yamagata et al., 2023) and Advantage Conditioned Transformer (ACT) (Gao et al., 2023) re-label the data trajectories using return-to-go or advantage obtained by dynamic programming. Elastic Decision Transformer (EDT) (Wu et al., 2023) adjusts the length of trajectories maintained in DT to facilitate the stitching of trajectories during testing. Liu & Abbeel (2023) prepare multiple trajectories sorted in ascending order by total rewards and re-labels them with the highest total reward in the trajectories before training DT. These efforts enhance the stitching capability, demonstrating that agents trained in offline RL can achieve better performance. While our focus is on model improvement and thus our scope differs from these efforts, we anticipate that combining these efforts and our model could improve the performance of both.

Several training methods have been proposed that emphasize the performance of DT agents when transitioning from offline training to online activity. Online Decision Transformer (Zheng et al., 2022) fine-tunes policies pre-trained on offline datasets through online interactions with the environment. Janner et al. (2021) add beam search to enable the agent to explore better actions. Wang et al. (2023) introduce a trained value function into DT, bridging the gap between the deterministic nature of return-conditioned supervised learning and the probabilistic characteristics of value-based methods. While these online enhancements diverge from the primary focus of this study, we believe that a combination of this study with these enhancements can potentially open up further possibilities.

6. Conclusion

We have explored the development of advanced, controllable agents in offline RL based on DT. Our proposed Return-Aligned Decision Transformer (RADT) separates the return sequence from the state-action sequence and can significantly improve the alignment of actual returns with the target return. The architectural innovations of RADT are a distinctive cross-attention mechanism and adaptive layer normalization for returns, designed to be easily integrated into the transformer block. In both MuJoCo and Atari domains, RADT demonstrated superior capability in aligning the actual return with the target return compared to existing DT-based methods. We believe that this capability allows us to obtain agents at various skill levels, and it broadens the range of applications for DT, such as in traffic analysis and as tools for games and education.

References

- Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *Proc. of ICML*, 2020.
- 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(1):253–279, 2013.
- 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. In *Proc. of NeurIPS*, 2021.
- David, S. B., Zimerman, I., Nachmani, E., and Wolf, L. Decision S4: Efficient sequence-based RL via state spaces layers. In *Proc. of ICLR*, 2023.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of ICLR*, 2021.
- Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline RL via supervised learning? In *Proc. of ICLR*, 2022.
- 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.
- Fujimoto, S. and Gu, S. A minimalist approach to offline reinforcement learning. In *Proc. of NeurIPS*, 2021.
- Gao, C., Wu, C., Cao, M., Kong, R., Zhang, Z., and Yu, Y. Act: Empowering decision transformer with dynamic programming via advantage conditioning. *arXiv preprint arXiv:2309.05915*, 2023.
- Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Proc. of NeurIPS*, 2021.
- Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline RL? In *Proc. of ICML*, 2021.
- Kim, J., Lee, S., Kim, W., and Sung, Y. Decision ConvFormer: Local filtering in MetaFormer is sufficient for decision making. In *Proc. of ICLR*, 2024.
- Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *Proc. of ICLR*, 2022.
- Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Proc. of NeurIPS*, 2020.
- 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.
- Liu, H. and Abbeel, P. Emergent agentic transformer from chain of hindsight experience. In *Proc. of ICML*, 2023.
- Nguyen, J., Powers, S. T., Urquhart, N., Farrenkopf, T., and Guckert, M. An overview of agent-based traffic simulators. *Transportation Research Interdisciplinary Perspectives*, 12:100486, 2021.
- Nguyen, V.-Q., Suganuma, M., and Okatani, T. Grit: Faster and better image captioning transformer using dual visual features. In *Proc. of ECCV*, 2022.
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.
- Schmidhuber, J. Reinforcement learning upside down: Don’t predict rewards—just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.
- Shang, J., Kahatapitiya, K., Li, X., and Ryoo, M. S. StAR-former: Transformer with state-action-reward representations for visual reinforcement learning. In *Proc. of ECCV*, 2022.
- Shen, R., Zheng, Y., Hao, J., Meng, Z., Chen, Y., Fan, C., and Liu, Y. Generating behavior-diverse game ais with evolutionary multi-objective deep reinforcement learning. In *Proc. of IJCAI*, 2021.
- Singla, A., Rafferty, A. N., Radanovic, G., and Heffernan, N. T. Reinforcement learning for education: Opportunities and challenges. *arXiv preprint arXiv:2107.08828*, 2021.
- Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.
- Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *Proc. of IROS*, 2012.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proc. of NeurIPS*, 2017.
- Wang, Y., Yang, C., Wen, Y., Liu, Y., and Qiao, Y. Critic-guided decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2312.13716*, 2023.
- Wu, Y.-H., Wang, X., and Hamaya, M. Elastic decision transformer. In *Proc. of NeurIPS*, 2023.

Xu, H., Jiang, L., Li, J., and Zhan, X. A policy-guided imitation approach for offline reinforcement learning. In *Proc. of NeurIPS*, 2022.

Yamagata, T., Khalil, A., and Santos-Rodriguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In *Proc. of ICML*, 2023.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *Proc. of ICML*, 2022.

A. Additional Results

A.1. Further Visualizations of Main Results

Concerning the results of Fig. 5 and Fig. 7 presented in Sec. 4.3, we illustrate the comparison of performance for each target return in Fig. 9. The black dotted line represents $y = x$, indicating that the actual return matches the target return perfectly. The closer to the black dotted line, the better the result. In the MuJoCo domain, it is shown that there is a correlation between the target return and the actual return for all methods. In contrast, in the Atari domain, especially in the Qbert task, the baseline shows a lack of correlation with the target return. Furthermore, there tends to be a performance that exceeds the black dotted line, suggesting that aligning with the target return is more difficult compared to the MuJoCo domain.

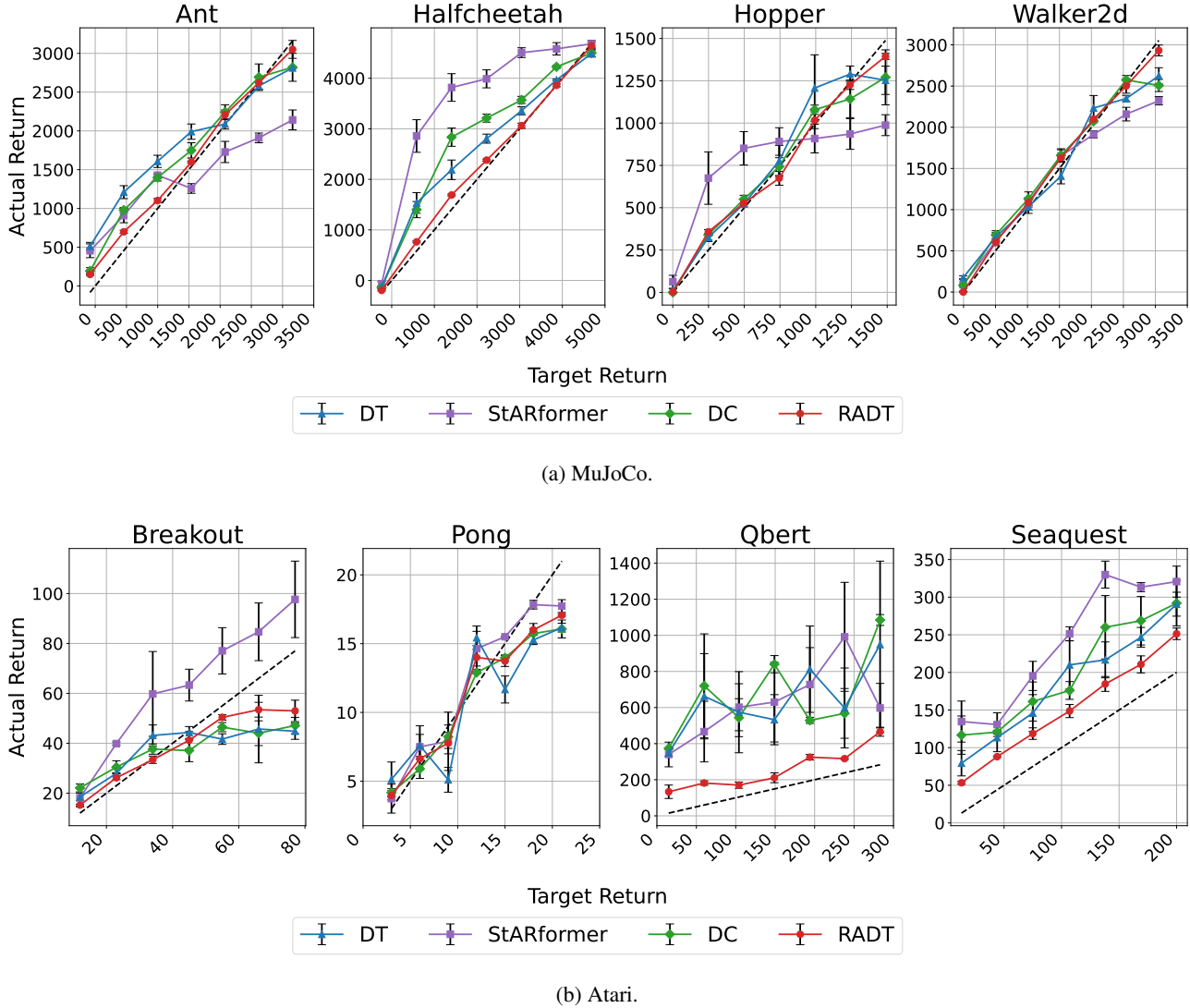


Figure 9. Comparisons of actual returns per target return in the experiments of Fig. 5 and Fig. 7. Each column represents a task. The x-axis represents the target return, and the y-axis represents the actual return. Target returns are set in the same way as Fig. 5 and Fig. 7. The black dotted line represents $y = x$, indicating that the actual return matches the target return perfectly. We report the mean and standard error over three seeds.

A.2. Scaling Parameters in Cross-Attention

We conduct experiments to evaluate the scaling parameter α , included in the cross-attention introduced in Sec. 3.2. The results are shown in Tab. 3. In these experiments, we did not employ adaptive layer normalization in either scenario, whether with or without α . We observed that the introduction of cross-attention without scaling improves the results for some tasks compared to DT. Furthermore, the inclusion of the scaling parameter α significantly enhance performance, surpassing DT in all tasks.

Table 3. **Experimental results evaluating the scaling parameters α in the MuJoCo domain.** CA represents DT with our cross-attention, and we prepare versions with and without the scaling parameters. The results of DT are also included.

Method	Scaling	Ant	HalfCheetah	Hopper	Walker
DT		2.69 $\pm$ 2.07	2.66 $\pm$ 2.07	2.19 $\pm$ 6.49	2.39 $\pm$ 2.60
CA		1.97 $\pm$ 0.13	2.74 $\pm$ 0.35	2.15 $\pm$ 0.10	2.90 $\pm$ 0.21
CA	✓	1.29 $\pm$ 0.38	1.51 $\pm$ 1.19	1.20 $\pm$ 1.55	1.24 $\pm$ 2.66

B. Experimental Details

B.1. Comparison of discrepancies

The discrepancies in Fig. 2 are calculated as the sum of the absolute errors between the actual return and target return across multiple tasks. The absolute error is normalized by the difference between the top 5% and bottom 95% of returns within the dataset.

B.2. Hyperparameter Settings

The full list of hyperparameters for RADT can be found in Tab. 4 and Tab. 5. We use the model code for DT, StAR-former, and DC from the following sources. DT: <https://github.com/kzl/decision-transformer>. StAR-former: <https://github.com/eliccassion/StARformer>. DC: <https://openreview.net/forum?id=af2c8EaKl8>.

Table 4. **Hyperparameters settings of RADT in the MuJoCo domain.**

Hyperparameter	Value
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity function	GELU, transformer SiLU, adaptive layer normalization
Context length K	20
Dropout	0.1
Learning rate	10^{-4}
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training steps
Position Encoding	Sinusoidal Position Encoding

Table 5. Hyperparameters settings of RADT in the MuJoCo domain.

Hyperparameter	Value
Number of layers	6
Number of attention heads	8
Embedding dimension	128
Batch size	512 Pong 128 Breakout, Qbert, Seaquest
Nonlinearity	ReLU encoder GELU transformer SiLU adaptive layer normalization
Encoder channels	32, 64, 64
Encoder filter size	$8 \times 8, 4 \times 4, 3 \times 3$
Encoder strides	4, 2, 1
Max epochs	5
Dropout	0.1
Learning rate	6×10^{-4}
Adam betas	(0.9, 0.95)
Grad norm clip	0.1
Weight decay	0.1
Learning rate decay	Linear warmup and cosine decay
Warmup tokens	$512 * 20$
Final tokens	$2 * 500000 * K$
Position Encoding	Sinusoidal Position Encoding