Self-Trained Decision Transformer

Dongzhuyuan Lu, Yening He
University of Waterloo
{d72lu, y37he}@uwaterloo.ca

Abstract

We introduce a framework that trains a Decision Transformer (DT) using a dataset with part of its data collected by the DT itself. We term this DT as a **S**elf-**T**rained **D**ecision-**T**ransformer. STDT enables a DT to train in an Offline Reinforcement Learning (RL) environment where data are scarce or existing data exhibit poor quality. Our project consists of three main parts: the first part involves reproducing the code from the original DT manuscript and applying it to **racetrack-v0** environment from Assignment 2, and the second part focuses on ways to improve DT, conducting extensive experiments on STDT as well as an improved version of DT, Elastic Decision Transformer (EDT). We found that DT does not perform well in the environment such as **hopper-medium-replay** as indicated in the origianl work, and STDT performs better than DT in **racetrack-v0** environment. Finally, we perform a comparative analysis and discussion on the improvements and their results. The codes are available here: `https://github.com/Tim-Lu-cuhksz/STDT`.

1 Introduction

In RL, an agent is trained to learn an optimal policy to achieve the maximum sum of returns [13]. With deep neural networks integrated into RL, it has demonstrated superior performance in various domains such as playing games [7, 12] and robotics [15]. One of the areas of RL is Offline RL [6] where an agent is trained using a sequence of pre-collected data rather than interacting with the environment in real-time. Recent advancements in transformers [14] have sparked a series of work to model RL as a sequence of states, actions, and rewards. As opposed to traditional RL methods which often rely on value function approximation and Bellman Backup [13], the novel transformer-based approach trains a transformer model, as widely used in language and vision tasks [3], on pre-collected experience via a sequence modeling objective, and this was adopted in DT [2]. DT feeds a sequence of states, actions, and rewards into a causal Transformer [9] that leverages masked self-attention layers to predict future actions. This architecture solves the credit assignment problem [8] by implicitly capturing the action-reward relationships through the Query-Key similarities embedded in the self-attention layers [14]. While traditional RL approaches often encounter challenges such as instabilities caused by bootstrapping in temporal difference methods [13], short-sighted decision making introduced by reward discounting [2], and difficulty in managing delayed or sparse rewards [2], DT aims to eliminate these obstacles by using a sequence modeling framework, leveraging autoregressive models to extract optimal policies from offline datasets without direct environment interaction. DT addresses several key challenges in RL including robustness to sparse or delayed reward signals and reliable learning in offline RL settings, using fixed datasets without additional exploration [2]. Our project reproduces their work on HighwayEnv (`https://github.com/Farama-Foundation/HighwayEnv`) to test and verify its robustness.

Several follow-up works have made an attempt to improve the original DT. One of them is the Elastic Decision Transformer (EDT), which takes a variable length of the traversed trajectory as the input so as to ensure the quality of the trajectory fed into the DT architecture [16]. It approximates a value maximizer and "stitch" a path of optimal length [16]. Unlike DT, which uses a fixed history length,

EDT dynamically adjusts the history length during inference based on the quality of the current trajectory, enabling more adaptive decision-making. EDT varies the history length during inference, allowing it to optimize decisions based on the quality of the current trajectory. It also uses *expectile regression*, which estimates the maximum achievable return for different history lengths, leading to improved action inference [16]. While DT uses a fixed history length, EDT's dynamic mechanism introduces flexibility, making it more effective in complex environments requiring trajectory stitching. We reproduce this work and conduct experiments on D4RL [4] and compare its results with the original DT.

We introduce the Self-Trained Decision Transformer (STDT) which is a DT trained on a collection of fixed-size and scarce Offline data, and then it becomes a data collector that generates trajectories that are then fed into itself for subsequent training. This approach is inspired by the self-play strategy adopted in [12] where a RL Go agent acts as its opponent to play against itself. In our project, however, the agent does not play against itself but with the environment. We expect that the DT could not only learn the old policy but surpass its performance from its interaction with the environment. Experiments have been conducted in the **racetrack-v0** environment from HighwayEnv to verify its effectiveness.

Our Contributions: 1) We use a pre-trained Proximal Policy Optimization (PPO) model [11, 10] to collect Offline RL data from the **racetrack-v0** environment which could be used in future work as a benchmark to test Offline RL methods. 2) We introduce Self-Trained Decision Transformer, a framework of Offline RL that addresses the challenge of scarce and poor Offline data. 3) Our experimental evaluation suggests that SDTD can achieve better performance than the original DT in **racetrack-v0** environment. We conduct experiments on EDT, an improved version of DT as well.

2 Preliminaries

2.1 Offline Reinforcement Learning

In this project, we consider training an agent in a Markov decision process MDP) described by $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$ where there is a sequence of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$. In a model-based RL setting, the world dynamics $P(s^{'}|s, a)$ and the reward function $r = \mathcal{R}(s, a)$ are given so that the agent can update its Q-values via Bellman Backup [13]. We characterize trajectories as a sequence of tokens which can be structured as follows:

$$\tau = (R_1, s_1, a_1, R_2, s_2, a_2, \ldots, R_T, s_T, a_T)$$

where R_t is the return-to-go, s_t represents the state, and a_t denotes the action at time step t. The ultimate goal is to train an agent that maximizes the expected (discounted) sum of returns.

In Offline RL, only a fixed-size dataset is available to train the agent and it cannot interact with the environment. In our setting, we are provided with a fixed-size dataset and able to interact with the environment as well. We would like to use the limited trajectories to train a preliminary model and generate new trajectories based on that. We repeat the process by appending the newly generated trajectories for future training.

2.2 Decision Transformer

Reward Representation Instead of learning from raw rewards, DT leverages *return-to-go* R_t, the cumulative sum of future rewards:

$$R_t = \sum_{t'=t}^{T} r_{t'}$$

This enables DT to condition its predictions on desired performance outcomes.

Model Architecture The last K timesteps of the target trajectory are fed into DT, yielding a total of $3K$ tokens. DT uses a Transformer-based structure with causal masking for autoregressive predictions. Inputs (states, actions, returns-to-go) are projected into high-dimensional embeddings, processed through the Transformer to predict future actions. Each layer is normalized [1] before being fed into the next layer.

Training and Loss Function The model is trained using the offline trajectory dataset. The loss is computed as:

$$\mathcal{L} = \frac{1}{K} \sum_{t=1}^{K} ||a_t - \hat{a}_t||^2$$

Where K is the context length and $\hat{a}_t$ is the predicted action.

2.3 Elastic Decision Transformer

EDT uses similar trajectory representation as in DT. However, EDT introduces a dynamic history length mechanism, allowing the trajectory length T to be optimized.

Dynamic History Length EDT determines the optimal history length T by solving:

$$T^* = \arg\max_T R_t(\tau_T)$$

where $R_t(\tau_T)$ represents the expected return for a given history length T. This enables EDT to prioritize high-quality segments of the trajectory.

Expectile Regression To estimate the maximum return R_t, EDT uses expectile regression:

$$\min_{R_t} \mathbb{E}\left[L_\alpha\left(R_t - \hat{R}_t\right)\right], \quad L_\alpha(x) = |\alpha - \mathbb{1}_{x<0}|x^2$$

where α is the expectile level, emphasizing higher returns over lower ones.

Training Objective EDT's training objective builds upon DT with an additional loss term for estimating maximum returns:

$$L_{\text{EDT}} = c_r L_{\text{return}} + L_{\text{state}} + L_{\text{action}} + L_{\text{max}}$$

where L_{max} represents the loss for expectile regression.

Inference During inference, EDT searches for the optimal history length T^* that maximizes R_t and truncates the trajectory to length T^* and predicts the next action. EDT's dynamic history adjustment improves adaptability and decision-making, while DT relies on a fixed history length.

3 Methodology

In this section, we present Self-Trained Decision Transformer (STDT), a framework that trains a DT using a limited and fixed dataset, and then new trajectories are generated by the pre-trained DT. We use the newly generated trajectories to "self-train" our DT model at the end of the iteration. Note that we only keep episodes of high quality (i.e., the reward should exceed a predefined threshold) to merge into the original dataset. We repeat this process to "self-train" the DT model and expect that DT could learn effective policies beyond what the first set of trajectories demonstrated.

Architecture We adopt the Generative-Pretrained Transformer (GPT) [9] as used in the original DT [2] manuscript. The model learns a linear layer for each state, action, and return-to-go modality, which are projected to the embedding layer. The feature vectors are passed through masked self-attention layers during training, and a future action token is predicted. Our proposed STDT is based on the GPT architecture to iteratively generate new trajectories. A basic framework is visualized in figure 1. The psuedo-code is also provided in algorithm 1.

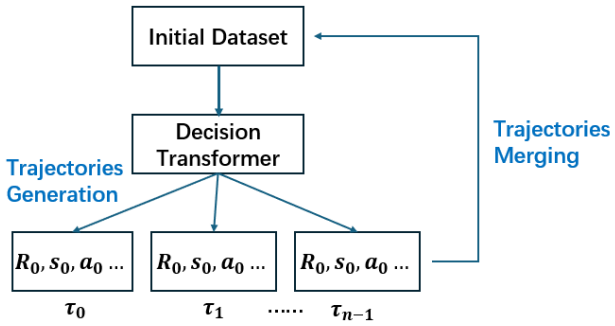

Figure 1: STDT Framework

Algorithm 1 Self-Trained Decision Transformer Pseudo-code

Input: $\mathcal{D}$, max_steps
Output: DT parameters θ_{DT}
Initialize parameters θ_{DT} for DT
$i \leftarrow 0$
for (R, s, a, t) in dataloader($\mathcal{D}$) **do**
  a_pred = DecisionTransformer(R, s, a, t)
  $loss = mean((a\_pred - a)^2)$
  Update θ_{DT} by minimizing $loss$ using Stochastic Gradient Descent (SGD)
  $\mathcal{D}_{gen} \leftarrow$ Generate new trajectories using the updated DT
  $\mathcal{D} \leftarrow$ merge($\mathcal{D}, \mathcal{D}_{gen}$)
  if $i \geq max\_steps$ **then** break
  end if
  $i \leftarrow i + 1$
end for
return θ_{DT}

Training The fixed dataset is generated by a PPO model trained in the **racetrack-v0** environment from HighwayEnv. We use PPO because **racetrack-v0** is continuous by default and PPO simplifies the policy optimization process by introducing a clipping mechanism to prevent excessive updates [11]. More technical details can be found in 4.3.

Evaluation As in [2], we specify a target return to STDT based on what we observe from the given dataset. For instance, we could use the maximum return among all trajectories from the initial dataset. After each update of the dataset, we re-compute the target reward again in case a better policy is learned.

4 Experiments

We first reproduce DT and EDT on D4RL environments in 4.1 to test whether the two algorithms are indeed effective as indicated in the original work. In 4.2, we test the original DT in **racdtrack-v0** environment and compare its results with PPO. In addition to that, we verify the effectiveness of STDT in **racetrack-v0** environment in 4.3.

4.1 Reproduction of DT & EDT on D4RL

We report the reproduction results (denoted by Repro.) and compare them with the original results in the table 4.1. We keep the setting the same as that in [2, 16] except that we reduce the number of training and evaluation rounds due to the limit of computational resources.

Fig 2 shows the training loss of mean action of EDT evaluated on D4RL environment. The training resutls of DT can be found in fig 7 to fig 12 in appendix A. We observe that our reproduction result

Dataset	DT	DT (Repro.)	EDT	EDT (Repro.)
hopper-medium	67.6 ± 1.0	58.0 ± 11.3	63.5 ± 5.8	58.2 ± 2.3
hopper-medium-replay	82.7 ± 7.0	66.4 ± 9.1	89.0 ± 8.3	87.8 ± 2.3
walker-medium	74.0 ± 1.4	68.6 ± 11.3	72.8 ± 6.2	70.2 ± 0.1
walker-medium-replay	66.6 ± 3.0	63.4 ± 12.5	74.8 ± 4.9	71.6 ± 0.6
halfcheetah-medium	42.6 ± 0.1	41.1 ± 11.2	42.5 ± 0.9	42.1 ± 0.6
halfcheetah-medium-replay	36.6 ± 0.8	35.9 ± 6.7	37.8 ± 1.5	37.1 ± 0.6

Table 1: Baseline comparisions on D4RL [4] tasks. DT is trained for 10 iterations and is evaluated by 100 episodes. EDT is trained for 500 iterations.

of DT in **hopper-medium-replay** is much lower than that in the original DT. However, this complies with the result in the manuscript of EDT [16]. Our reproduction results indicate a higher variance evaluated by DT. We believe that if we increase the number of evaluation rounds, the variance would be as low as that in DT. The training loss of mean return and state of EDT can be found in fig 13 and 14 in appendix A. The figures indicates that the loss of mean action, return, and state decreases gradually across different datasets. The mean action loss converges after 100 training steps.

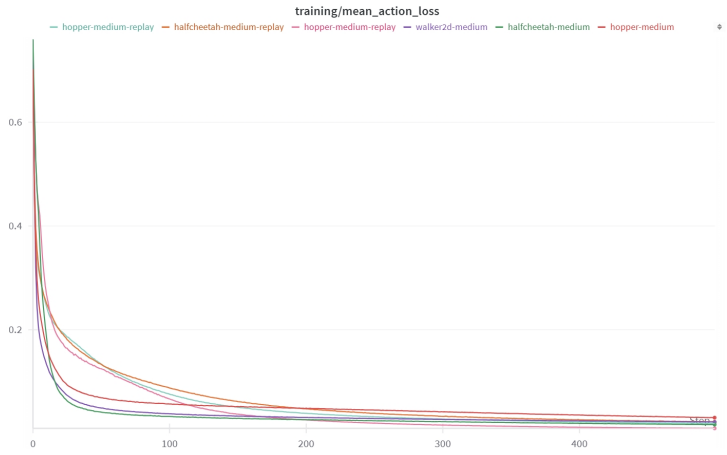

Figure 2: Mean Action Loss of EDT on D4RL

We also compare the performance of EDT across different datasets when the number of training steps reaches 250 and 500 as depicted in fig **??** in appendix A. In general, EDT performs better with more training steps.

hopper DT shows generally better performance than the traditional Offline RL methods such as Conservative Q-learning [5]. EDT shows strong performance on the replay datasets, effectively leveraging its trajectory-stitching capability to enhance decision-making.

walker2d The results indicate that EDT demonstrates its strengths on replay datasets, where its dynamic history length mechanism is highly effective for handling sub-optimal data. However, its improvement is less pronounced in other environments, such as **halfcheetah**, suggesting room for further refinement in the mechanism.

halfcheetah While the **halfcheetah** environment demonstrates limited improvement with EDT in the original results, the significant discrepancies in reproduced results highlight potential differences in experimental conditions. All in all, EDT demonstrates significant improvements over DT, especially in replay datasets, showcasing its ability to effectively handle sub-optimal trajectories. However, the observed deviations in certain reproduced results highlight the need for careful setup and validation in experiments. Overall, EDT's dynamic mechanisms provide a robust framework for enhancing offline reinforcement learning tasks.

4.2 Reproduction of DT on HighwayEnv

We train a PPO policy via Stable-Baselines3 [10] in **racetrack-v0** environment and use it to generate 256 episodes as our Offline dataset. We test the effectiveness of DT in **racetrack-v0** from HighwayEnv and we increase the difficulty level by adding more vehicles on the road as shown in fig 3. Specifically, we set the number of vehicles to 5 on the road and enable lateral actions only. Due to limited computational resources, we decrease the number of training steps of DT from 10,000 to 1,000. We keep other settings the same as that in DT.

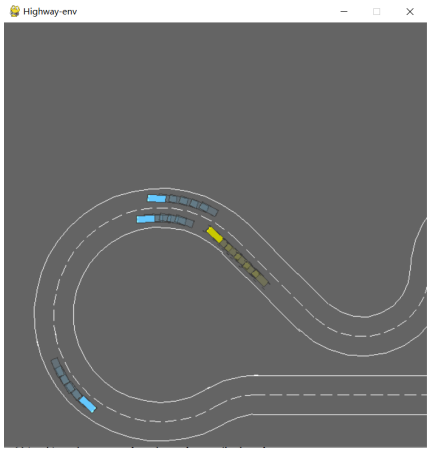

Figure 3: **racetrack-v0** Environment

Since the gradient is based on the action loss as in [2], we show the training action error of DT in **racetrack-v0**. Fig 4 shows that the action error decreases over time, indicating that DT is learning to behave as similarly as the PPO model that generates the dataset.

Figure 4: Training Error of DT in **racetrack-v0** Environment

We set the target return to 1500 as the maximum return in an episode in the Offline dataset is 1507. The mean return evaluated after training for 10 iterations is shown in fig 5. The mean return is around 250 during evaluation. The return mean fluctuates during training as well, indicating that DT fails (crashes) quickly in some episodes. But in general, DT outperforms the PPO policy (achieves a mean return of about 150 after 40,000 timesteps of training as indicated in fig 15 in appendix A), which is used to generate our Offline dataset.

4.3 STDT on HighwayEnv

We adopt the same Offine dataset as that used in evaluating DT. The number of epochs we use for training STDT is set to 5, each of which contains 4 training iterations. We generate 15 episodes in each epoch and use 500 as the training steps, different from DT where we use 1,000. As before, the return-to-go is set to 1500. Fig 6 presents the evaluation result of STDT on **racetrack-v0**. The mean return of STDT is about 350 after training while that of DT achieves only around 250, indicating the

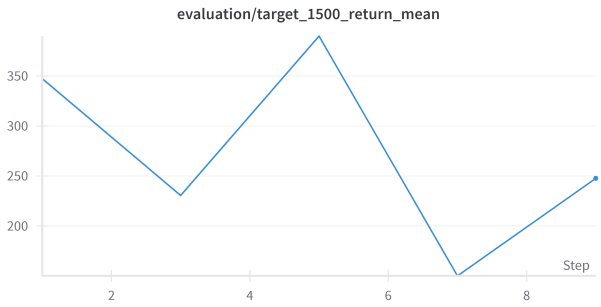

Figure 5: Mean Return of DT in **racetrack-v0** Environment

effectiveness of STDT. The training result can be found in fig 16 in appendix A. We also observe that the performance of STDT fluctuates as in 6, potentially due to the quality of newly generated trajectories.

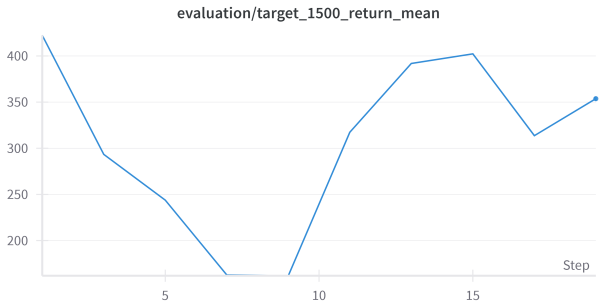

Figure 6: Mean Return of STDT in **racetrack-v0** Environment

5 Discussions & Limitations

DT demonstrates superior or comparable performance over the state-of-the-art Offline RL methods. However, we observe that DT is to some extent limited by the quality of the Offline dataset. In our experiments, DT learns well following the Offline policy by struggles to learn better behavior. We believe that more training steps are necessary to reveal more meaningful patterns.

EDT represents a significant improvement over DT by introducing a dynamic history length mechanism. This innovation enables more effective trajectory stitching and superior performance in offline RL tasks, bridging the gap between DT and Q-learning-based methods. By addressing DT's limitations, EDT sets a new benchmark for sequence modeling in reinforcement learning. Our experiment results show that EDT performs better but to limited extent.

STDT performs better than the original DT by a small margin, which to some extend shows the effectiveness of STDT. In qualitative analysis, however, we found that STDT does not overcome what the PPO policy has suffered[1]. For example, overtaking vehicles while remaining on track appears to be a challenge for both the PPO policy and STDT.

Limitations 1) Even though STDT generates new trajectories to "self-train" itself, the trajectories may lack diversity as compared to the original dataset. In other words, STDT suffers from generating meaningfully novel trajectories that tackles the challenges faced by the policy used to generate the Offline dataset. 2) Since we only enable lateral actions, the agent could not handle the situations where both lanes are occupied by other vehicles, leading to collisions. 3) Due to limited computational resources, we are only able to train DT and STDT on HighwayEnv for a small number of epochs,

[1]Videos can be found on our Github repositories: `https://github.com/Tim-Lu-cuhksz/STDT`

which may not reveal meaningful patterns. In addition to that, the number of episodes we generate in each epoch is limited as well.

6 Conclusions & Future Work

In this project, we introduce the framework of STDT and reproduce the code from the original DT and EDT work and test their effectiveness on D4RL environments. We found out that our reproduction result of DT on **hopper-medium-replay** appears to be much lower than that in DT but aligns with the result in EDT paper. We also applies DT on **racetrack-v0** from HighwayEnv and observes that DT manages to learn the PPO policy used to generate the Offline dataset effectively in less training steps than trained in D4RL environment. Built upon DT, STDT demonstrates better performance on **racetrack-v0** but is constrained by the ability of PPO as well.

Future work could potentially address the above-mentioned issues of STDT by introducing more exploration when generating the new trajectories. More experiments should be conducted on **racetrack-v0** by increasing the number of training epochs, the episodes generated in each epoch, and enabling longitudinal actions to test the robustness of STDT.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. arXiv:1607.06450 [stat].

[2] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

[4] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning, February 2021. arXiv:2004.07219 [cs].

[5] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

[6] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, November 2020. arXiv:2005.01643 [cs].

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. arXiv:1312.5602 [cs].

[8] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A Survey of Temporal Credit Assignment in Deep Reinforcement Learning, July 2024. arXiv:2312.01072 [cs].

[9] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training.

[10] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[12] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[13] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[15] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning Generalizable Dexterous Manipulation from Human Grasp Affordance. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 618–629. PMLR, December 2023.

[16] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic Decision Transformer. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 18532–18550. Curran Associates, Inc., 2023.

A Appendix / supplemental material

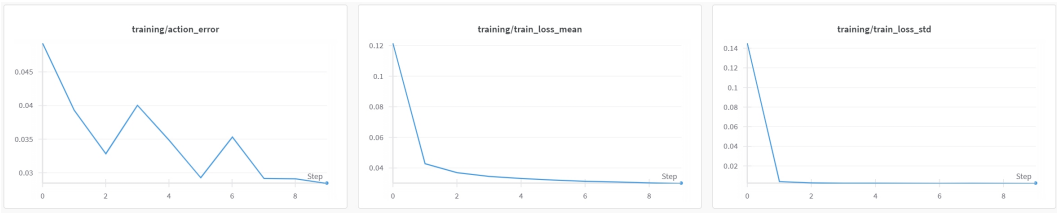

Figure 7: Training Loss of DT on **halfcheetah-medium**

Figure 8: Training Loss of DT on **halfcheetah-medium-replay**

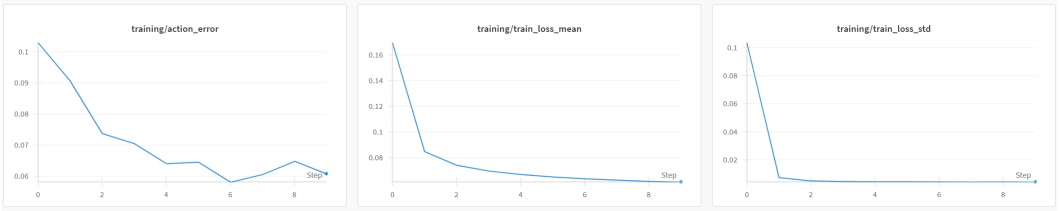

Figure 9: Training Loss of DT on **hopper-medium**

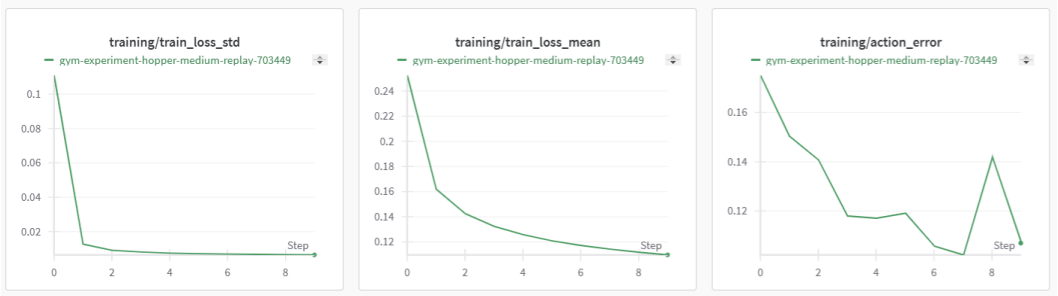

Figure 10: Training Loss of DT on **hopper-medium-replay**

Figure 11: Training Loss of DT on **walker2d-medium**

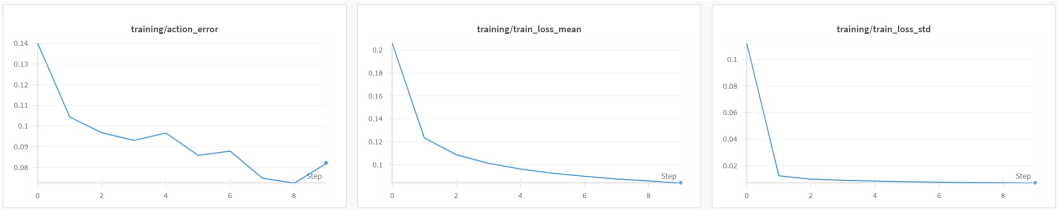

Figure 12: Training Loss of DT on **walker2d-medium-replay**

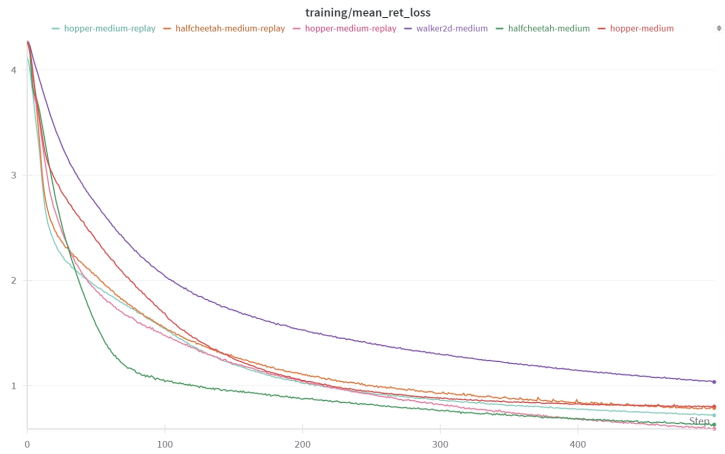

Figure 13: Mean Return Loss of EDT on D4RL

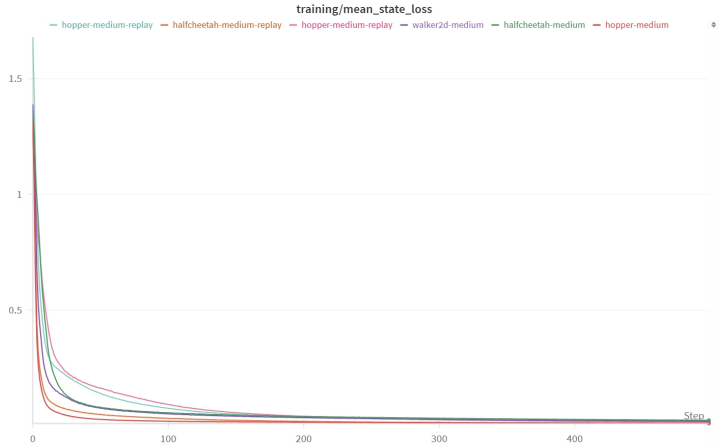

Figure 14: Mean State Loss of EDT on D4RL

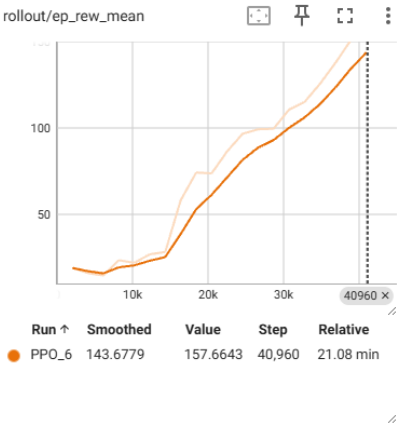

Figure 15: Mean Return of PPO in **racetrack-v0** Environment

Figure 16: Mean Action Error of STDT in **racetrack-v0** Environment