Target Return Optimizer for Multi-Game Decision Transformer

Kensuke Tatematsu T.KEN1358@AKANE.WASEDA.JP
Waseda University, Japan

Akifumi Wachi AKIFUMI.WACHI@LYCORP.CO.JP
LY Corporation, Japan

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Achieving autonomous agents with robust generalization capabilities across diverse games and tasks remains one of the ultimate goals in AI research. Recent advancements in transformer-based offline reinforcement learning, exemplified by the Multi-Game Decision Transformer (Lee et al., 2022), have shown remarkable performance across various games or tasks. However, these approaches depend heavily on human expertise, presenting substantial challenges for practical deployment, particularly in scenarios with limited prior game-specific knowledge. In this paper, we propose an algorithm called Multi-Game Target Return Optimizer (MTRO) to autonomously determine game-specific target returns within the Multi-Game Decision Transformer framework using solely offline datasets. MTRO addresses the existing limitations by automating the target return configuration process, leveraging environmental reward information extracted from offline datasets. Notably, MTRO does not require additional training, enabling seamless integration into existing Multi-Game Decision Transformer architectures. Our experimental evaluations on Atari games demonstrate that MTRO enhances the performance of RL policies across a wide array of games, underscoring its potential to advance the field of autonomous agent development.

Keywords: Offline Reinforcement Learning, Decision Transformer, Target Return

1. Introduction

Reinforcement learning (RL) involves an agent interacting with an environment to learn a policy to maximize the expected cumulative reward. RL is a powerful framework for solving sequential decision-making problems and has been applied to various applications such as alignment of large language models (Ouyang et al., 2022), recommendation systems (Afsar et al., 2022; Chen et al., 2019), and robotics (Plappert et al., 2018; Wu et al., 2023). Offline RL (Levine et al., 2020) is a paradigm for learning policies from a pre-collected dataset, which is particularly effective when environmental interaction is challenging due to cost or safety concerns. Despite promising developments in various approaches to offline RL, it is typically applied to solve a single task within the same environment. Recently, with advances in foundational models (Bommasani et al., 2021), research in multi-game RL has progressed, aiming to apply a single policy model to various game environments characterized by different state-transition functions or reward functions. This approach has been gaining interest as it seeks to acquire a general policy that can adapt to diverse environments from a variety of datasets.

Decision Transformer (DT, Chen et al. (2021)) and its variants leverage the inference capabilities of transformer models (Vaswani, 2017) by treating offline RL as a sequence modeling problem. This approach improves performance while addressing stability challenges associated with long-term credit assignments (Sutton and Barto, 2018). Multi-Game DT extends the original DT to multi-game RL settings, enabling robust decision-making. Furthermore, similar to the scaling laws of large language models (Hoffmann et al., 2022; Kaplan et al., 2020), significant improvements have been observed over traditional offline RL methods such as conservative Q-learning (CQL, Kumar et al. (2020)) and behavior cloning (Pomerleau, 1991).

In both DT and Multi-Game DT, the learning process involves inputting the desired return into the transformer to learn a policy that generates conditionally actions. This framework allows control over the desired performance by adjusting the target return input during inference. However, determining the optimal target return to generate the best actions is challenging. In typical usage of the DT, the target return is manually set by humans. Additionally, Multi-Game DT addresses this by assuming a distribution over expert actions to generate optimal behaviors. While these methods are effective in a limited number of game environments, it becomes impractical to determine target returns based on human knowledge for multiple tasks in real-world scenarios.

This paper proposes an algorithm called Multi-Game Target Return Optimizer (MTRO). This algorithm automatically determines game-specific target returns from offline datasets, without relying on human knowledge. This method enhances the performance of pre-trained Multi-Game DT models across multiple games without additional training. Our contributions are summarized as follows:

- We propose MTRO that consists of the following two components, which enable us to optimize game-specific target returns from offline datasets:
  1. Data-driven Expert Return Distribution (DERD) that automatically determines distributions related to expert actions, replacing human assumptions.
  2. Bayes-Adjusted Return Prediction (BARP) that compares the predicted return distribution of the transformer with offline datasets and adjusts it based on predictive performance.
- We empirically demonstrate that MTRO improves performance in multi-game RL environments by obtaining game-specific target returns.

2. Related Work

Decision Transformer (DT, Chen et al. (2021)) is a promising approach that frames RL as a sequence modeling problem using the transformer architecture (Vaswani, 2017). DT demonstrates high performance due to the inference capabilities and scalability of the Transformer architecture, leading to extensive research in this area. Recent studies focus on advancements in the Transformer architecture and its representation methods (Shang et al., 2022; Kim et al., 2023; Furuta et al., 2021; Janner et al., 2021), as well as the integration of conventional RL techniques such as dynamic programming (Yamagata et al., 2023; Gao et al.,

2024). Beyond offline RL settings, research has explored fine-tuning DT online after pre-training offline (Zheng et al., 2022; Meng et al., 2021) and generalizing to unknown tasks from limited offline demonstration datasets (Xu et al., 2022, 2023). Additionally, there is growing interest in incorporating safety considerations into RL (Liu et al., 2023).

DT in multi-game RL. DT architectures have also been proven effective in multi-game RL settings. Multi-Game DT (Lee et al., 2022) extends the standard DT to multi-game RL settings, extracting knowledge from large volumes of offline datasets. However, the diverse nature and complexity of games make creating a generalized policy challenging, necessitating the exploration of various approaches. HarmoDT (Hu et al., 2024) identifies the optimal parameter space for each task, while Elastic DT (Wu et al., 2024) proposes a method to switch from suboptimal to optimal trajectories by adjusting the history length input to DT.

Target return in DT. In the DT paradigm, setting a target return as a condition for the transformer facilitates the generation of future actions that achieve the desired return. The target return is an important element in determining the performance of actions and has been extensively studied. Q-learning DT (Yamagata et al., 2023) utilizes the outcomes of dynamic programming to relabel target returns in offline datasets for learning. Additionally, Multi-Game DT estimates target returns that generate expert actions by employing Bayes' rule during testing. Return-Aligned DT (Tanaka et al., 2024) not only aims to improve performance through target returns but also seeks to mitigate discrepancies between target returns and actual returns.

3. Preliminaries

We consider a sequential decision-making problem in which an agent interacts with multiple environments. Each environment is modeled as a partially observable Markov decision process (POMDP), which is defined as a tuple

$$\mathcal{M} \coloneqq \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{Z}, r \rangle, \tag{1}$$

where $\mathcal{S} \coloneqq \{s\}$ is a set of states, $\mathcal{A} \coloneqq \{a\}$ is a set of actions, $\mathcal{O} \coloneqq \{o\}$ is a set of observations, $\mathcal{P}$ is a state transition probability, $\mathcal{Z}$ is the observation density function meaning the probability of receiving observation $o_t \in \mathcal{O}$ from state $s_t \in \mathcal{S}$, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. At each time step t, the agent in state $s_t \in \mathcal{S}$ receives an observation $o_t \sim \mathcal{Z}(\cdot \mid s_t)$ from the environment, selects an action $a_t \in \mathcal{A}$, and then receives a reward $r_t = r(s_t, a_t)$ and transits to a next state $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$. Return of a trajectory R_t is defined as the cumulative sum of rewards from the time step t to the final step T, expressed as $R_t = \sum_{t'=t}^{T} r_{t'}$.

This paper aims to learn a single policy π_θ parameterized by θ that maximizes the future return R_t in all environments where each environment is associated with each game.

3.1. Decision Transformer (DT)

DT (Chen et al., 2021) is a paradigm that treats an RL problem as a sequence modeling problem, using the transformer architecture to derive effective policies from offline datasets.

Unlike traditional RL methods (Mnih et al., 2015a; Mnih, 2016; Schulman et al., 2017) that depend on value functions or policy gradients, this approach predicts future actions based on past sequence information. In DT, a trajectory is defined as follows:

$$\tau = (R_1, o_1, a_1, R_2, o_2, a_2, ..., R_T, o_T, a_T). \tag{2}$$

The DT model is trained on offline datasets to predict action a_{t+1} based on a history of previous return $R_{\leq t}$, observation $o_{\leq t}$, and action $a_{\leq t}$. During testing, the agent infers a using observation o obtained from the environment and the target return $\hat{R}$ specified by a human.

3.2. Multi-Game Decision Transformer

While standard DTs aim to solve a single game, the Multi-Game DT (Lee et al., 2022) explores multi-game settings, and aims to enable a single DT model to solve multiple games through learning from offline datasets. In Multi-Game DT, a sequence is defined as

$$\tau = (o_1, R_1, a_1, r_1, o_2, R_2, a_2, r_2, \ldots, o_T, R_T, a_T, r_T). \tag{3}$$

Unlike the standard DT, Multi-Game DT utilizes information about the immediate reward r. The model's inputs are $o_{\leq t}$, $R_{\leq t}$, $a_{\leq t}$, and $r_{\leq t}$, while the outputs are R_{t+1}, a_{t+1}, and r_{t+1}. The prediction of future returns R_{t+1} and rewards r_{t+1} is conducted to acquire better representations. Additionally, the prediction of the return R is used to set the target return $\hat{R}$ for outputting expert action sequences, as discussed in the next section.

3.3. Determining the Target Return

The target return is vital for determining the quality of the generated action sequence. In DT, the interaction begins with a human specifying the desired performance as the initial target return $\hat{R}_1$ for each game. The target return $\hat{R}_t$ is iteratively updated at each step using the received reward r_t by

$$\hat{R}_{t+1} = \hat{R}_t - r_t, \quad \forall t \in [1, T-1]. \tag{4}$$

Empirical experiments in the original DT paper (Chen et al., 2021) have shown a strong correlation between the target return and the actual observed episode return.

In DT, the training data was limited to those generated by medium-level or expert policies. However, in Multi-Game DT, non-expert data is utilized alongside expert data to learn a general-purpose agent. Thus, Multi-Game DT utilizes Bayes' rule with the return prediction distribution $P(R_t \mid \ldots)$ from the transformer to automatically generate the target return $\hat{R}$. The target return $\hat{R}$ is sampled from the expert return distribution $P(R_t, \ldots \mid \text{expert}_t)$ using the Bayes' rule:

$$P(R_t, \ldots \mid \text{expert}_t) \propto P(\text{expert}_t \mid R_t, \ldots) P(R_t \mid \ldots), \tag{5}$$

where $P(\text{expert}_t \mid R_t, \ldots)$ is a probability that the behavior is expert-level. Existing studies, as represented by Multi-Game DT (Lee et al., 2022), typically assume a binary classifier using an inverse temperature κ, with R_{low} and R_{high} as the lower and upper bounds of the return, which is defined as:

Algorithm 1 Action Generation by MTRO

$P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots) \leftarrow \frac{n^{R_t}_{\text{expert}}}{N(R_t)}$
for $t = 1, 2, \ldots$ **do**
  if $t < \ell$ **then**
    $\alpha(R_t) \leftarrow 1$
  else if $t = \ell$ **then**
    $\alpha(R_t) \leftarrow D_{\text{KL}}\left(\frac{1}{\ell}\sum_{t'=1}^{\ell} P(R_{t'} \mid \ldots) \,\middle\|\, \frac{N(\tilde{R}_1)}{N_{\text{total}}}\right) + 1$
  end if
  $\log P(R_t, \ldots \mid \text{expert}_t) \leftarrow \log P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots) + \frac{1}{\alpha(R_t)} \cdot \log P(R_t \mid \ldots) + \left(1 - \frac{1}{\alpha(R_t)}\right) \cdot \log \frac{N(\tilde{R}_1)}{N_{\text{total}}}$
  Normalize $P(R_t, \ldots \mid \text{expert}_t)$
  Sample a target return $\hat{R} \sim P(R_t, \ldots \mid \text{expert}_t)$
  Sample a next action $a_t \sim P(a_t \mid \hat{R}_t, \ldots)$
end for

$$P(\text{expert}_t \mid R_t, \ldots) \propto \exp\left(\kappa\left(\frac{R_t - R_{\text{low}}}{R_{\text{high}} - R_{\text{low}}}\right)\right). \tag{6}$$

This assumption implies that the likelihood of a return sequence being classified as expert increases with larger target return $\hat{R}_t$. In other words, higher target returns are associated with more expert-like behavior sequences. This formulation enables the model to sample and set a target return $\hat{R}$ that is expected to lead to high performance.

Challenges of the conventional method. However, this method has limitations when it comes to improving performance across different games. Applying Bayes' rule requires that the assumption $P(R_t, \ldots \mid \text{expert}_t)$ matches the actual environment. Thus, using a single assumption across multiple environments with diverse reward structures, as in (6), poses challenges. For example, in one game getting 100 points is easy for casual players. In another game, even expert players may find 100 points hard to reach. Assuming that a higher score always indicates expert play overlooks the fact that each game has its own scoring system and difficulty. Addressing these limitations could greatly enhance the effectiveness of multi-game RL policies across different environments.

4. Method

This paper proposes an algorithm called Multi-Game Target Return Optimizer (MTRO). A key feature of MTRO is that the probability $P(\text{expert}_t \mid R_t, \ldots)$ is derived directly from offline datasets, rather than relying on human assumptions as in (6). This allows for improved performance across games without human knowledge, regardless of the reward structure of each game. The pseudo-code of MTRO is outlined in Algorithm 1.

In this paper, we apply MTRO under an assumption that a pre-trained Multi-Game DT model is available a priori. The main objective of MTRO is to improve performance in multiple games without additional training by determining more appropriate target returns for Multi-Game DT models.

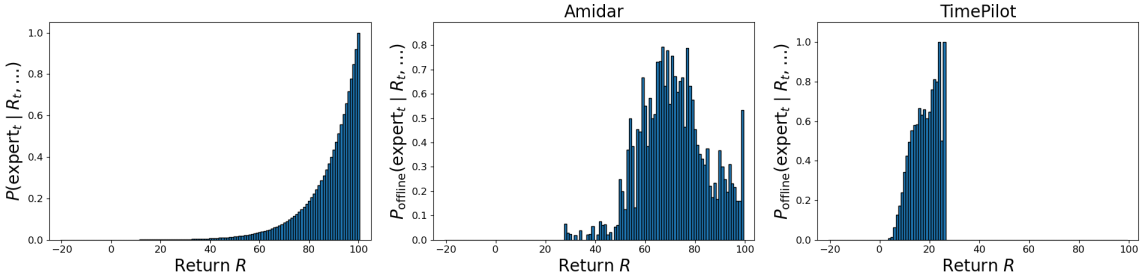

Figure 1: This figure compares distributions $P(\text{expert}_t \mid R_t \ldots)$ derived from (6) and (7). **Left**: $P(\text{expert}_t \mid R_t, \ldots)$ under the assumption of (6), which increases exponentially from R_{low} to R_{high} for the quantized returns in the range [-20, 100]. **Middle**: $P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots)$ based on (7) for the Amidar game, where the distribution is closer to a normal distribution rather than an exponential type. **Right**: $P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots)$ based on (7) for the TimePilot game, showing a distribution closer to an exponential type, but the maximum return value is significantly smaller than that in the left figure.

4.1. DERD: Data-driven Expert Return Distribution

Traditionally, Multi-Game DT has applied the standard Bayes' rule (5) based on the assumption represented as (6). This equation suggests that the probability of being an expert increases as the return approaches R_{high}. However, R_{high} defines the maximum episode return across all games, but not all games have the episode in the offline datasets that reaches R_{high}. Furthermore, due to the various reward structures present in different games, the distribution of $P(\text{expert}_t \mid R_t, \ldots)$ may not naturally conform to an exponential form.

To address these challenges, we calculate $P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots)$ using episodic return information from offline datasets. In this approach, the target return can be set individually for each game, reflecting the unique reward structures of each game; thus, we can enhance the probability of generating expert-level state action sequences within the context of specific games. Specifically, each return is quantized to an integer value $\tilde{R}$, allowing us to approximately compute $P(\text{expert}_t \mid R_t, \ldots)$ as follows:

$$P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots) \propto \frac{n_{\text{expert}}^{\tilde{R}_t}}{N(\tilde{R}_t)}, \tag{7}$$

where $n_{\text{expert}}^{\tilde{R}_t}$ represents the number of episodes with quantized return $\tilde{R}_t$ labeled as expert data, and $N(\tilde{R}_t)$ represents the number of episodes with return $\tilde{R}_t$.

This formula represents the probability that a quantized return $\tilde{R}_t$ is classified as expert within the offline datasets. By calculating it for each game, this approach aims to capture the different game-specific reward structures, enhancing the accuracy of expert classification. This method is simple and easy to implement, as it requires calculating the probability only once before testing, and then simply replacing it with (6).

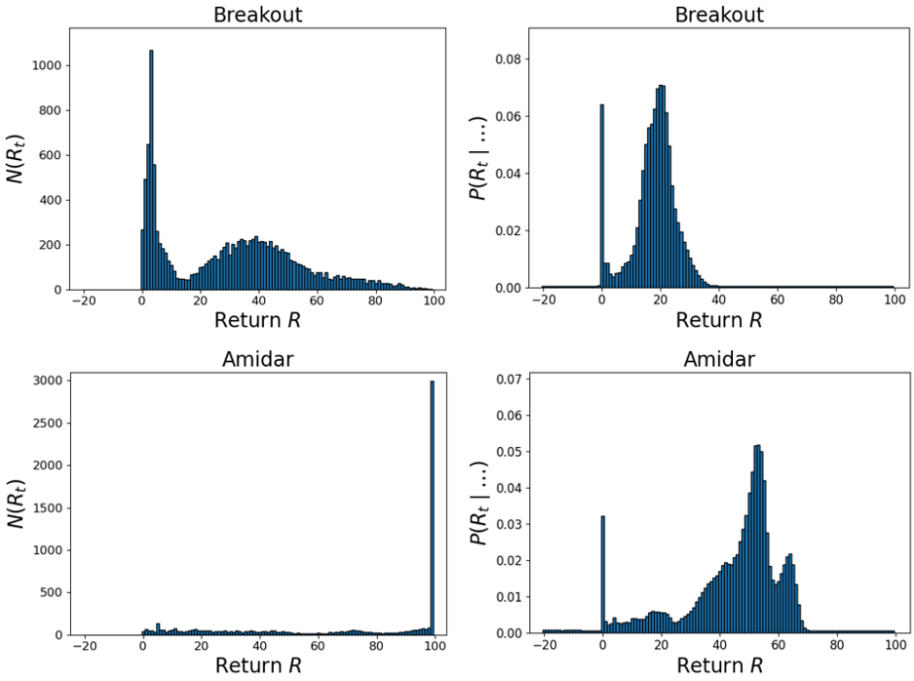

Figure 2: This figure illustrates the differences between $N(R_1)$ derived from offline datasets, and the transformer's output $P(R_t \mid \ldots)$. **Top**: In the Breakout Game, $P(R_t \mid \ldots)$ tends to lean towards lower R values compared to $N(R_t)$, suggesting that lower returns are more probable than what the offline datasets indicates. **Bottom**: In the Amidar Game, while $N(R_t)$ peaks at higher R values, $P(R_t \mid \ldots)$ shows a notably different pattern, highlighting a divergence from the expected distribution.

Figure 1 compares the results of calculations from (6) and (7). Figure 1 (left) shows the values computed using (6), which is used in all games in the Multi-Game DT. On the other hand, Figure 1 (Middle, Right) presents the results derived from (7) based on offline datasets, highlighting the differences from the exponential type. Our method allows for the use of $P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots)$ that considers the difficulty and reward structure of each game.

4.2. BARP: Bayes-Adjusted Return Prediction

In Multi-Game DT, which learns multiple games with different reward designs using a single transformer model, it is also challenging to accurately predict the distribution of returns $P(R_t \mid \ldots)$ from the model's output. When predictions of $P(R_t \mid \ldots)$ are inaccurate, we cannot generate valid target returns in (5) due to inaccurate $P(R_t, \ldots \mid \text{expert}_t)$. Therefore,

we modify (5) while incorporating the prediction accuracy of returns as follows:

$$\begin{aligned}\log P(R_t, \ldots \mid \text{expert}_t) \propto \log P(\text{expert}_t \mid R_t, \ldots) \\ + \frac{1}{\alpha(R_t)} \cdot \log P(R_t \mid \ldots) \\ + \left(1 - \frac{1}{\alpha(R_t)}\right) \cdot \log \frac{N(\tilde{R}_t)}{N_{\text{total}}},\end{aligned} \tag{8}$$

where N_{total} represents the total number of episodes in the offline datasets, and $N(\tilde{R}_t)/N_{\text{total}}$ represents the probability distribution over the offline datasets for each $\tilde{R}_t$. The function $\alpha(R_t)$ is defined based on the KL-divergence between the average of the predicted distribution $P(R_{t=0\sim\ell} \mid \ldots)$ from the first few ℓ frames during inference and $N(\tilde{R}_t)/N_{\text{total}}$ with respect to R_t. While we use $\alpha(R_t) = 1$ for all $t < \ell$, we define $\alpha(R_t)$ for all $t \geq \ell$ as follows:

$$\alpha(R_t) \coloneqq D_{\text{KL}}\left(\frac{1}{\ell}\sum_{t=1}^{\ell} P(R_t \mid \ldots) \middle\| \frac{N(\tilde{R}_1)}{N_{\text{total}}}\right) + 1. \tag{9}$$

When $\alpha(R_t) = 1$ (i.e., the KL-divergence is equal to 0), the predicted return distribution $P(R_t \mid \ldots)$ is considered accurate, and the standard Bayes' rule (5) is applied. If the KL-divergence is greater than 0, $P(R_t \mid \ldots)$ is adjusted by adding the distribution of the target return from the offline datasets, $N(\tilde{R}_t)/N_{\text{total}}$, in (8).

Figure 2 compares $N(R_t)$ calculated from offline datasets with $P(R_t \mid \ldots)$ obtained during the inference of a Multi-Game DT model. The significant discrepancy between these distributions indicates poor predictive accuracy of $P(R_t \mid \ldots)$, suggesting that Bayes' theorem does not hold. We address this issue by using (8).

5. Experiment

In this paper, we conduct empirical evaluations to assess the performance of our MTRO. First, we evaluate its performance across various games in comparison with other baselines. Next, we show that the two components of MTRO (i.e., DERD and BARP) lead to more reasonable target return distributions $P(R_t \mid \text{expert}_t)$ than simply using the Bayes' rule (5).

5.1. Problem Settings

We evaluate MTRO using the Atari benchmark (Bellemare et al., 2013; Gulcehre et al., 2020) as with the Multi-Game DT. In this paper, we use the DQN Replay Dataset (Agarwal et al., 2020) as offline datasets. This comprehensive dataset comprises Atari game trajectories where rewards are clipped to $[-1, 1]$ and the returns are quantized to the range $\{-20, 100\}$. It includes 50 million steps of experience for each of the five DQN agents (Mnih et al., 2015a) recorded during their learning processes. In our experiments, we focus on 39 of the 41 games learned by Multi-Game DT, for which the score achieved by a human player is publicly available.

5.2. Metrics

The performance of individual Atari games is measured using the Human Normalized Score (HNS) (Mnih et al., 2015b), calculated with the following formula:

$$\text{HNS} = \frac{\text{score} - \text{score}_{\text{random}}}{\text{score}_{\text{human}} - \text{score}_{\text{random}}}, \tag{10}$$

where $\text{score}_{\text{random}}$ is the score obtained by playing the game with random actions and $\text{score}_{\text{human}}$ is the score achieved by a human player (Quan and Ostrovski, 2020). To create an aggregate comparison metric across all games, we use the Interquartile Mean (IQM, Agarwal et al. (2021)) of the human normalized scores across all games. The IQM is computed after conducting 50 trials for each game, with a single model.

5.3. Implementation of MTRO

We implemented MTRO based on the publicly available pre-trained model of Multi-Game DT[1], which has been trained on 41 Atari games and consists of 200 million parameters. By adding two key techniques of DERD and BARP, we improved performance without additional training.

Specifically, for implementing DERD, we calculated $N(R_t)$ and $n^{R_t}_{\text{expert}}$ from offline datasets. This involved utilizing experience from a single DQN agent within the DQN replay dataset, as used in Multi-Game DT. We categorized the data for each game, designating the final 10% episodes of the experience as expert level. Additionally, we computed the KL-divergence as in (9). For this, we saved $P(R_t \mid \ldots)$ from the first few steps during inference. In our experiments, we set $\ell = 20$, using the first 20 inference steps. Starting from the 21st step, we applied (8) to determine the target return. Thus, MTRO incurs additional computation only during the first few steps of inference, and its computational cost is not significantly different from that of Multi-Game DT.

5.4. Baseline Methods

We also implemented four baseline methods.

Multi-Game DT. Multi-Game DT utilizes 100 million steps of experience from two agents, derived from the DQN Replay Dataset, for each of the 41 games. This dataset encompasses actions from all stages of learning, containing both expert and non-expert data. Consequently, when aiming to generate expert-like behavior, Multi-Game DT applies Bayes' theorem (5).

Naïve method. As another baseline method, we utilize a straightforward approach using

$$\begin{aligned}\log P(R_t, \ldots \mid \text{expert}_t) \propto \log P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots) \\ + \frac{1}{\alpha(R_t)} \cdot \log P(R_t \mid \ldots).\end{aligned} \tag{11}$$

This method serves as an alternative to correcting $P(R_t \mid \ldots)$ with (8). The function $\alpha(R_t)$ is defined as with (8). When $\alpha(R_t)$ is 1 (i.e., the KL-divergence is equal to 0), the standard

1. https://sites.google.com/view/multi-game-transformers

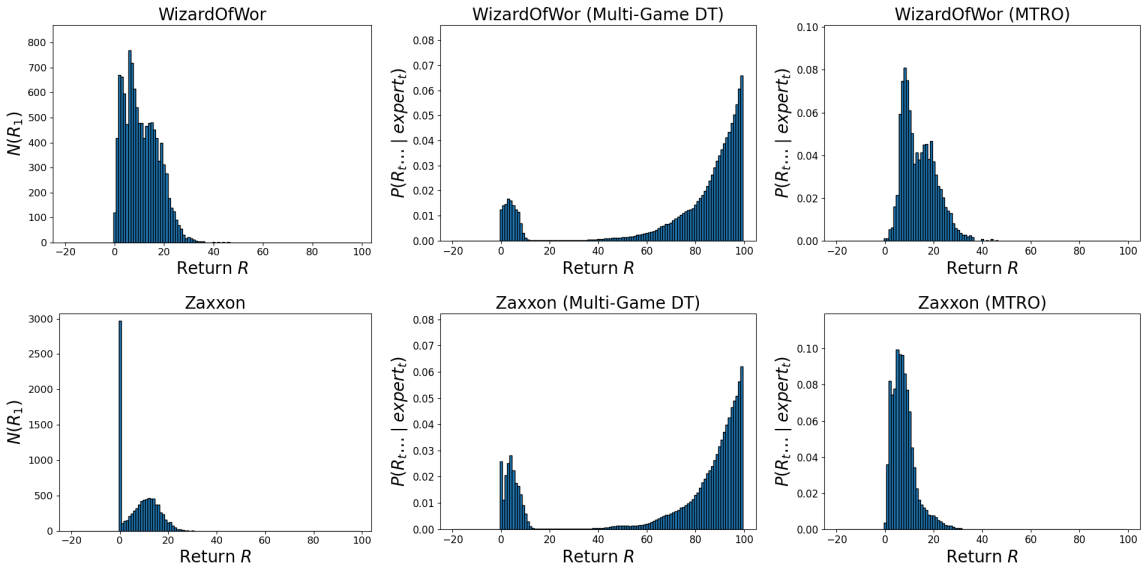

Figure 3: This figure demonstrates the effectiveness of target returns in MTRO by comparing the frequency of episode returns from offline datasets, $N(R_1)$, with $P(R_t, \ldots \mid \text{expert}_t)$ used for sampling target returns in both Multi-Game DT and MTRO. Here, $P(R_t, \ldots \mid \text{expert}_t)$ represents the distribution at the beginning of episodes in the games. In both games, it is evident that Multi-Game DT might select target returns in the range $R = 60 \sim 100$, which are not included in $N(R_1)$. On the other hand, because MTRO samples feasible target returns based on offline datasets, it is suitable as input to the transformer and is expected to generate expert actions.

Bayes' rule is applied. If $\alpha(R_t)$ is greater than 1, we address the prediction accuracy issue of $P(R_t \mid \ldots)$ by increasing the influence of $P_{\text{offline}}(\text{expert}_t \mid R_t, \ldots)$. We can also interpret that (11) is a simplified version of MTRO where the last term of (8) is ignored.

Two baselines for ablation studies. We also implement two baseline methods where either DERD and BARP is removed from MTRO in order to show that their combinations are needed to enhance the performance of MTRO.

5.5. Main Results

We compare the performance of MTRO and four baseline methods to investigate the effectiveness of MTRO. Table 1 shows the average HNS for each method, along with the mean, the standard deviation, and the overall IQM. MTRO, using both DERD and BARP, achieves the highest IQM score with 28% improvement over Multi-Game DT. Notably, the Naïve method and MTRO (w/o BARP), both using DERD, also demonstrate strong performance, confirming the effectiveness of DERD. Table 3, provided as supplementary material, summarizes the raw scores.

Table 1: Results for the Atari games, including the Human Normalized Score (HNS) with mean, standard deviation, and overall IQM over 50 trials. Bold text indicates the highest score for each game and the highest IQM score for the algorithm.

Game Name	Multi-Game DT	Naïve method	MTRO(w/o BARP)	MTRO(w/o DERD)	MTRO
Amidar	0.081 ± 0.046	0.093 ± 0.047	0.095 ± 0.051	**0.097 ± 0.045**	0.080 ± 0.030
Assault	3.626 ± 1.647	3.901 ± 1.749	3.732 ± 1.788	**4.003 ± 1.745**	3.463 ± 1.768
Asterix	0.831 ± 0.232	0.998 ± 0.312	**1.019 ± 0.486**	0.871 ± 0.250	0.984 ± 0.376
Atlantis	3.935 ± 9.934	**15.547 ± 19.076**	**15.547 ± 19.076**	-0.274 ± 0.201	14.769 ± 19.657
BankHeist	-0.009 ± 0.013	-0.010 ± 0.012	**-0.001 ± 0.021**	-0.008 ± 0.018	-0.010 ± 0.012
BattleZone	0.346 ± 0.194	0.425 ± 0.170	**0.451 ± 0.209**	0.368 ± 0.191	0.425 ± 0.178
BeamRider	0.461 ± 0.286	0.462 ± 0.257	0.463 ± 0.252	**0.468 ± 0.244**	0.451 ± 0.268
Boxing	7.600 ± 0.484	7.595 ± 0.478	7.580 ± 0.538	**7.677 ± 0.427**	7.675 ± 0.418
Breakout	8.494 ± 2.864	10.363 ± 2.894	10.592 ± 3.360	8.387 ± 3.767	**11.306 ± 2.756**
Centipede	0.023 ± 0.167	0.064 ± 0.158	**0.082 ± 0.193**	0.077 ± 0.186	0.054 ± 0.148
ChopperCommand	**0.147 ± 0.139**	0.101 ± 0.136	0.102 ± 0.136	0.048 ± 0.115	0.087 ± 0.132
CrazyClimber	4.408 ± 0.762	4.618 ± 0.772	4.618 ± 0.772	**4.698 ± 0.711**	4.618 ± 0.772
DemonAttack	**6.920 ± 4.035**	5.764 ± 3.116	5.910 ± 3.464	6.175 ± 3.687	6.390 ± 5.460
DoubleDunk	2.182 ± 2.161	1.564 ± 2.329	**2.545 ± 2.294**	1.027 ± 1.960	1.345 ± 2.349
Enduro	1.502 ± 0.321	1.620 ± 0.373	1.620 ± 0.373	**1.661 ± 0.313**	1.620 ± 0.373
FishingDerby	**1.689 ± 0.648**	1.603 ± 0.697	1.475 ± 0.738	1.573 ± 0.734	1.517 ± 0.739
Freeway	1.009 ± 0.057	1.003 ± 0.053	1.005 ± 0.053	0.845 ± 0.065	**1.010 ± 0.052**
Frostbite	**0.497 ± 0.201**	0.129 ± 0.145	0.108 ± 0.134	0.341 ± 0.199	0.190 ± 0.149
Gopher	4.460 ± 2.397	5.321 ± 3.027	4.229 ± 2.591	5.309 ± 3.166	**5.550 ± 3.822**
Gravitar	**-0.035 ± 0.040**	-0.041 ± 0.034	-0.040 ± 0.034	-0.041 ± 0.029	-0.042 ± 0.033
Hero	0.608 ± 0.085	0.629 ± 0.078	0.643 ± 0.044	**0.648 ± 0.031**	0.647 ± 0.032
IceHockey	0.152 ± 0.275	0.274 ± 0.289	**0.286 ± 0.288**	0.043 ± 0.331	0.218 ± 0.319
Jamesbond	1.713 ± 0.981	**2.389 ± 0.812**	2.122 ± 0.743	1.315 ± 1.049	2.235 ± 0.742
Kangaroo	**3.933 ± 0.963**	3.871 ± 1.107	3.812 ± 1.181	1.699 ± 1.314	3.763 ± 0.989
Krull	1.019 ± 0.733	-0.288 ± 0.462	0.985 ± 0.931	**6.378 ± 1.225**	6.348 ± 1.303
KungFuMaster	0.662 ± 0.287	**0.706 ± 0.269**	0.618 ± 0.227	0.663 ± 0.259	0.642 ± 0.220
NameThisGame	0.940 ± 0.269	0.943 ± 0.292	0.943 ± 0.292	**1.032 ± 0.272**	0.943 ± 0.292
Phoenix	0.649 ± 0.075	0.655 ± 0.065	0.643 ± 0.135	**0.663 ± 0.076**	0.638 ± 0.074
Qbert	0.444 ± 0.239	0.381 ± 0.200	0.339 ± 0.222	**0.467 ± 0.229**	0.458 ± 0.251
Riverraid	0.846 ± 0.219	0.817 ± 0.233	0.843 ± 0.209	0.864 ± 0.218	**0.893 ± 0.219**
RoadRunner	3.607 ± 1.060	3.360 ± 1.011	3.106 ± 1.144	3.221 ± 0.802	**3.775 ± 1.237**
Robotank	4.860 ± 1.034	4.854 ± 0.998	**4.934 ± 0.901**	4.324 ± 1.433	4.695 ± 1.220
Seaquest	**0.110 ± 0.037**	0.094 ± 0.051	0.069 ± 0.043	0.052 ± 0.043	0.090 ± 0.041
TimePilot	-0.632 ± 1.032	**-0.195 ± 0.986**	-0.502 ± 0.978	-0.555 ± 0.991	-0.448 ± 0.984
UpNDown	1.230 ± 0.564	1.172 ± 0.483	1.013 ± 0.489	1.156 ± 0.564	**1.256 ± 0.562**
VideoPinball	-11.419 ± 0.283	-11.324 ± 0.391	-11.374 ± 0.340	**-11.253 ± 0.570**	-11.296 ± 0.509
WizardOfWor	-0.049 ± 0.091	**-0.035 ± 0.100**	-0.050 ± 0.067	-0.044 ± 0.087	-0.042 ± 0.081
YarsRevenge	0.243 ± 0.169	0.246 ± 0.172	0.225 ± 0.175	0.236 ± 0.155	**0.283 ± 0.207**
Zaxxon	0.022 ± 0.046	0.014 ± 0.027	0.005 ± 0.016	0.003 ± 0.017	**0.030 ± 0.038**
IQM	1.35	1.59	1.61	1.37	**1.73**

To further evaluate the effectiveness of MTRO, we compared the performance of each method against the Multi-Game DT on a per-game basis. We focused on identifying games where performance improved or declined significantly. Table 2 shows that MTRO consistently achieves high scores across various games, demonstrating fewer declines and more improvements compared to other methods.

The reason for the improved performance by MTRO is to set feasible target returns based on offline datasets, unlike Multi-Game DT which utilizes distributions significantly different from the features of offline datasets. In Figure 3, we compare the distribution $P(\text{expert}_t \mid R_t, \ldots)$ for sampling target returns during game execution in both Multi-Game DT and MTRO, with the distribution of episode returns from offline datasets $N(R_t)$. MTRO

Table 2: Comparison of the number of games with performance improvements and declines across different methods. This table shows the number of games where each method achieves more than 10% higher or lower IQM compared to the Multi-Game DT.

Method	Games Improved	Games Declined
Naïve method	12	7
MTRO (w/o DERD)	8	12
MTRO (w/o BARP)	12	9
MTRO	14	4

sample valid target returns that align with the distribution $N(R_1)$. In other words, Multi-Game DT may encounter issues in action inference because it inputs returns not included in the training data to the Transformer. However, MTRO can promote expert actions by focusing on feasible target returns.

6. Conclusion

This paper proposes an algorithm called Multi-Game Target Return Optimizer (MTRO) to compute optimal target returns for each task within the Multi-Game Decision Transformer framework. Traditionally, determining target returns required applying human knowledge for each task. By leveraging environmental reward information from offline datasets, MTRO addresses the challenge of identifying suitable target returns necessary for generating expert actions, eliminating the reliance on human expertise. Furthermore, MTRO refines the expert distribution using offline datasets based on the prediction accuracy of the decision transformer. Our approach enhances the performance of pre-trained models through several modifications without requiring additional training. Extensive experiments on Atari games have confirmed that MTRO is effective across a diverse range of games.

Limitations. Our approach necessitates the selection of expert data from extensive offline datasets to compute the target return required for generating expert actions. This precondition is frequently met in numerous real-world robotics tasks, and offline RL algorithms commonly rely on high-quality demonstrations. However, a potential limitation arises in scenarios where expert data is unavailable for specific tasks within the offline datasets. Additionally, while this paper considers the experiences from the final stages of a DQN agent's training as expert data, there is room for discussion on the method used to determine how to qualify as expert data.

Future work. Our approach is not limited to the multi-game settings and has the potential to enhance the performance of the Decision Transformer framework more broadly. Exploring its application in various offline reinforcement learning algorithms based on Decision Transformers could yield valuable insights and improvements.

In this study, we focused on generating expert actions. However, with minor modifications to MTRO, it would be possible to generate medium-level actions as well. Future

work could explore these modifications to expand the range of behaviors that can be effectively generated, thereby broadening the applicability and utility of our approach in diverse reinforcement learning scenarios.

References

M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pages 104–114. PMLR, 2020.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:237353084.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

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. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*, pages 1052–1061. PMLR, 2019.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Rui Kong, Zongzhang Zhang, and Yang Yu. Act: Empowering decision transformer with dynamic programming via advantage conditioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12127–12135, 2024.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.18080*, 2024.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. *arXiv preprint arXiv:2310.03022*, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning*, pages 21611–21630. PMLR, 2023.

Linghui Meng, Muning Wen, Yaodong Yang, Chenyang Le, Xiyun Li, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, and Bo Xu. Offline pre-trained multi-agent decision transformer: One big sequence model tackles all smac tasks. *arXiv preprint arXiv:2112.02845*, 2021.

Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King,

Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015a. URL https://api.semanticscholar.org/CorpusID:205242740.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015b. URL https://api.semanticscholar.org/CorpusID:205242740.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL http://github.com/deepmind/dqn_zoo.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jinghuan Shang, Kumara Kahatapitiya, Xiang Li, and Michael S Ryoo. Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In *European conference on computer vision*, pages 462–479. Springer, 2022.

R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN 9780262039246. URL https://books.google.co.jp/books?id=sWV0DwAAQBAJ.

Tsunehiko Tanaka, Kenshi Abe, Kaito Ariu, Tetsuro Morimura, and Edgar Simo-Serra. Return-aligned decision transformer. *arXiv preprint arXiv:2402.03923*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. In *Conference on Robot Learning*, pages 618–629. PMLR, 2023.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. *arXiv preprint arXiv:2304.08487*, 2023.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pages 38989–39007. PMLR, 2023.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

Appendix A. Raw Atari Game Scores

Table 3: Results for the Atari games, including the raw score with mean, standard deviation over 50 trials.

Game Name	Multi-Game DT	Naïve method	MTRO (w/o BARP)	MTRO (w/o DERD)	MTRO
Amidar	144.82 ± 78.09	165.24 ± 80.70	168.78 ± 87.09	172.06 ± 77.03	143.04 ± 50.83
Assault	2106.66 ± 855.90	2249.40 ± 908.68	2161.32 ± 929.25	2302.24 ± 906.58	2021.68 ± 918.48
Asterix	7101.00 ± 1926.80	8486.00 ± 2584.72	8658.00 ± 4034.60	7435.00 ± 2076.30	8370.00 ± 3119.89
Atlantis	76506.00 ± 160720.03	264378.00 ± 308614.53	264378.00 ± 308614.53	8424.00 ± 3253.59	251788.00 ± 318011.62
BankHeist	7.80 ± 9.44	6.60 ± 8.86	13.80 ± 15.48	8.40 ± 13.62	7.00 ± 8.54
BattleZone	14400.00 ± 6764.61	17160.00 ± 5927.43	18080.00 ± 7282.42	15160.00 ± 6640.36	17160.00 ± 6197.94
BeamRider	8003.60 ± 4741.12	8022.48 ± 4256.97	8034.84 ± 4181.35	8116.00 ± 4033.65	7828.24 ± 4444.54
Boxing	91.30 ± 5.80	91.24 ± 5.74	91.06 ± 6.45	92.22 ± 5.13	92.20 ± 5.02
Breakout	246.32 ± 82.49	300.16 ± 83.35	306.74 ± 96.76	243.24 ± 108.50	327.30 ± 79.38
Centipede	2319.92 ± 1662.38	2729.88 ± 1569.62	2908.16 ± 1915.28	2859.08 ± 1843.11	2629.30 ± 1466.57
ChopperCommand	1778.00 ± 913.96	1474.00 ± 897.06	1484.00 ± 891.37	1126.00 ± 755.73	1386.00 ± 867.64
CrazyClimber	121208.00 ± 19079.16	126452.00 ± 19331.00	126452.00 ± 19331.00	128466.00 ± 17821.96	126452.00 ± 19331.00
DemonAttack	12738.60 ± 7340.09	10635.60 ± 5667.18	10902.20 ± 6300.36	11384.50 ± 6706.21	11774.30 ± 9930.65
DoubleDunk	-13.80 ± 4.75	-15.16 ± 5.12	-13.00 ± 5.05	-16.34 ± 4.31	-15.64 ± 5.17
Enduro	1292.62 ± 276.32	1393.82 ± 320.71	1393.82 ± 320.71	1429.28 ± 269.60	1393.82 ± 320.71
FishingDerby	-2.20 ± 34.35	-6.74 ± 36.97	-13.50 ± 39.10	-8.32 ± 38.90	-11.30 ± 39.16
Freeway	29.86 ± 1.67	29.68 ± 1.55	29.74 ± 1.57	25.02 ± 1.93	29.90 ± 1.53
Frostbite	2189.20 ± 857.12	616.60 ± 617.98	528.20 ± 572.80	1519.60 ± 848.26	876.20 ± 636.09
Gopher	9869.20 ± 5165.01	11724.40 ± 6522.97	9370.80 ± 5582.47	11698.40 ± 6823.07	12218.00 ± 8236.31
Gravitar	62.00 ± 127.89	42.00 ± 109.25	45.00 ± 108.28	44.00 ± 92.00	40.00 ± 104.40
Hero	19142.80 ± 2538.41	19766.60 ± 2333.44	20173.40 ± 1322.83	20339.90 ± 927.05	20319.40 ± 958.41
IceHockey	-9.36 ± 3.33	-7.88 ± 3.50	-7.74 ± 3.49	-10.68 ± 4.00	-8.56 ± 3.86
Jamesbond	498.00 ± 268.51	683.00 ± 222.40	610.00 ± 203.47	389.00 ± 287.28	641.00 ± 203.15
Kangaroo	11784.00 ± 2873.84	11598.00 ± 3302.03	11424.00 ± 3524.23	5120.00 ± 3919.29	11278.00 ± 2950.21
Krull	2686.00 ± 782.78	1291.00 ± 493.70	2649.60 ± 993.43	8406.80 ± 1307.64	8375.00 ± 1391.07
KungFuMaster	15136.00 ± 6454.48	16134.00 ± 6035.81	14144.00 ± 5111.48	15152.00 ± 5831.03	14690.00 ± 4955.81
NameThisGame	7705.20 ± 1549.92	7722.60 ± 1681.26	7722.60 ± 1681.26	8230.80 ± 1567.91	7722.60 ± 1681.26
Phoenix	4968.40 ± 486.19	5004.20 ± 420.69	4926.60 ± 876.75	5057.40 ± 490.61	4898.60 ± 482.68
Qbert	6065.00 ± 3179.56	5222.50 ± 2655.90	4670.00 ± 2951.43	6371.50 ± 3043.21	6257.00 ± 3341.90
Riverraid	14690.40 ± 3461.94	14236.20 ± 3671.93	14648.00 ± 3292.59	14973.60 ± 3435.19	15432.00 ± 3456.39
RoadRunner	28264.00 ± 8304.04	26334.00 ± 7923.17	24340.00 ± 8965.02	25240.00 ± 6282.23	29580.00 ± 9690.94
Robotank	49.34 ± 10.03	49.28 ± 9.68	50.06 ± 8.74	44.14 ± 13.90	47.74 ± 11.84
Seaquest	4689.20 ± 1542.62	4008.20 ± 2124.38	2971.40 ± 1805.58	2267.40 ± 1825.26	3866.60 ± 1730.78
TimePilot	2518.00 ± 1713.91	3244.00 ± 1637.21	2734.00 ± 1624.02	2646.00 ± 1645.50	2824.00 ± 1634.20
UpNDown	14261.60 ± 6293.27	13609.80 ± 5384.90	11841.60 ± 5461.39	13435.40 ± 6291.46	14550.60 ± 6268.76
VideoPinball	144.10 ± 399.25	278.40 ± 551.28	208.38 ± 479.88	378.40 ± 803.62	318.34 ± 718.69
WizardOfWor	358.00 ± 380.57	418.00 ± 418.42	354.00 ± 280.86	380.00 ± 364.97	388.00 ± 340.38
YarsRevenge	15625.96 ± 8724.59	15733.74 ± 8848.26	14683.52 ± 8988.26	15266.96 ± 7967.17	17672.94 ± 10656.70
Zaxxon	232.00 ± 422.58	164.00 ± 243.93	76.00 ± 142.21	60.00 ± 154.92	306.00 ± 349.52