In-Context Decision Transformer: Reinforcement Learning via Hierarchical Chain-of-Thought

Sili Huang [1 2] Jifeng Hu [1] Hechang Chen [* 1] Lichao Sun [3] Bo Yang [* 2]

Abstract

In-context learning is a promising approach for offline reinforcement learning (RL) to handle online tasks, which can be achieved by providing task prompts. Recent works demonstrated that in-context RL could emerge with self-improvement in a trial-and-error manner when treating RL tasks as an across-episodic sequential prediction problem. Despite the self-improvement not requiring gradient updates, current works still suffer from high computational costs when the across-episodic sequence increases with task horizons. To this end, we propose an In-context Decision Transformer (IDT) to achieve self-improvement in a high-level trial-and-error manner. Specifically, IDT is inspired by the efficient hierarchical structure of human decision-making and thus reconstructs the sequence to consist of high-level decisions instead of low-level actions that interact with environments. As one high-level decision can guide multi-step low-level actions, IDT naturally avoids excessively long sequences and solves online tasks more efficiently. Experimental results show that IDT achieves state-of-the-art in long-horizon tasks over current in-context RL methods. In particular, the online evaluation time of our IDT is **36**× times faster than baselines in the D4RL benchmark and **27**× times faster in the Grid World benchmark.

1. Introduction

Large transformer models (Vaswani et al., 2017) have shown impressive abilities across a variety of domains, including text (Brown et al., 2020b), image (Dosovitskiy et al., 2020), and audio (Alayrac et al., 2022). In the field of reinforcement learning (RL), large transformer models can treat the RL tasks as a type of sequential prediction problem, which has proven successful in using solely offline training (Lee et al., 2022; Reed et al., 2022). A notable shortcoming lies with these methods to self-improve when employed in online environments. To overcome this, in-context RL methods have been introduced, which enable continued policy improvement (Laskin et al., 2023).

Recent works demonstrated that in-context RL can automatically improve its performance in a trial-and-error manner when across-episodic contexts serve as prompt conditions (Lee et al., 2023a). The construction of the across-episodic context is flexible and easy to implement, such as a chain of experience that consists of multiple historical trajectories arranged in ascending order of returns (Hao Liu, 2023). Despite the progress made, current methods are mostly limited to short-horizon tasks with less than 100 timesteps (Laskin et al., 2023). This arises from (1) the quadratic complexity of the self-attention mechanism and (2) the significant increase in the length of sequences caused by across-episodic contexts. Such huge computational costs severely limit in-context RL to apply the trial-and-error ability on traditional RL tasks, which often reach 1000 timesteps, such as MuJoCo (Fu et al., 2020) and Atari (Bellemare et al., 2013).

In fact, trial-and-error is the central idea of modern RL algorithms (Sutton & Barto, 2018). It is an animal behavior originated by a psychologist Thorndike (1927) who considers trial-and-error as an elementary way of combining search and memory. Correspondingly, the across-episodic contexts provide memory, and the self-attention mechanism reviews historical actions in the memory to search for better actions. However, human decision-making is more complex and operates on multiple levels of temporal abstraction (Sutton et al., 1999b). For example, travelers tend to decide on their budget first, then their mode of transportation, right down to the smallest action. Inspired by this idea, a natural perspective emerges:

"Can human multi-level decision-making bring out a more efficient trial-and-error?"

As one high-level decision can guide multi-step low-level

[*]Equal contribution [1]School of Artificial Intelligence, Jilin University, China [2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, China [3]Lehigh University, Bethlehem, Pennsylvania, USA. Correspondence to: Bo Yang <ybo@jlu.edu.cn>, Hechang Chen <chenhc@jlu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

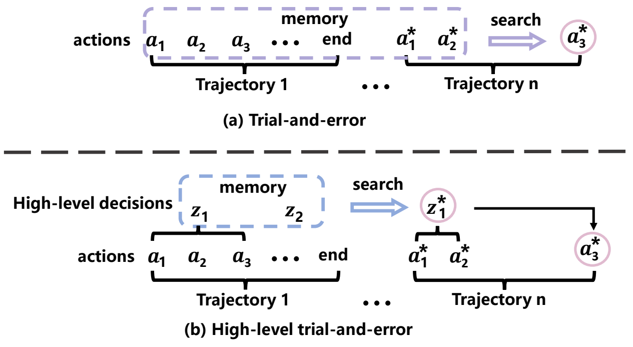

Figure 1. Trial-and-error comparison of minimal actions and high-level decisions, where * denotes better results. (a) In the trial-and-error process, the memory consists of the smallest actions from experiences and serves as context to search for better action. (b) In the high-level trial-and-error process, the memory and search act on high-level decisions. Since one high-level decision controls multiple actions, we can use smaller memory to preserve experiences and search for better decisions with less computational costs.

actions, it can considerably shorten across-episodic contexts and, therefore, significantly alleviate the computational costs. In this view, we aim to explore a high-level trial-and-error process, as shown in Figure 1. However, since the model generates high-level decisions rather than low-level actions that interact with the environment, our first challenge is ensuring that better high-level decisions can encourage better low-level actions. In addition, the in-context RL is trained with supervised losses, which predicts each next step conditioned on the past steps in the sequence. Unlike low-level actions, the high-level decision is an abstract concept that is usually not directly observable from training data.

To this end, we propose an efficient in-context RL method called In-context Decision Transformer (IDT). Specifically, IDT consists of Making Decisions, Decisions to Go, and Reviewing Decisions modules to mimic the high-level trial-and-error process. First, the Making Decisions module is a decoder-only transformer that generates high-level decisions autoregressively, where the high-level decision is represented by a vector sampled from a multivariate Gaussian distribution. Then, the generated high-level decisions are fed to the Decisions to Go module, which is also a decoder-only transformer to generate low-level actions autoregressively. The output of the Making Decisions module serves as a conditional input to the Decisions to Go module, ensuring high-level decisions correctly guide low-level actions. To fill in the missing high-level decisions in the training data, we designed the Reviewing Decisions module to encode high-level decisions from sequences of low-level actions. All three modules are learned end-to-end by predicting the low-level actions from the training data.

Our contributions are as follows: (1) We propose IDT, an in-context RL method that emerges with high-level trial-and-error ability. IDT can learn by directly combining sub-optimal data and efficiently improving itself through multiple trials at test time. (2) Compared to the contexts consisting of the smallest actions, IDT significantly shortens the evaluation time by **36**× times on the D4RL baselines (Fu et al., 2020) and **27**× times on the Large Grid World environments (Lee et al., 2022). (3) IDT can achieve state-of-the-art results with less training costs, especially outstanding in long-horizon tasks.

2. Related Work

Transformer for Decision-Making. In general, reinforcement learning was proposed as a fundamentally online paradigm (Sutton et al., 1999a). The nature of online learning comes with some limitations when meeting the applications for which it is impossible to gather online data and learn simultaneously, such as autonomous driving. To this end, offline RL proposes that the agent can learn from a fixed dataset of previously collected data without gathering new data during learning (Fujimoto et al., 2019; Kumar et al., 2020; Yu et al., 2021; Kumar et al., 2019; Hu et al., 2022; 2023). In the context of offline RL, recent works explored using transformer-based policy by treating RL tasks as a type of sequential prediction problem. Among them, a decision transformer (Chen et al., 2021) is proposed to model trajectories as sequences and autoregressively predicts action conditioning on desired return-to-go, past states, and actions. Trajectory transformer (Janner et al., 2021) demonstrated that transformer could learn single-task policies from offline data. Subsequently, the multi-game decision transformer (Lee et al., 2022) and Gato (Reed et al., 2022) further showed that transformer-based policies could address multi-tasks in the same domain and cross-domain tasks. However, these works focused on distilling expert policies from offline data and failed to enable self-improvement like IDT. Instead, when the offline data are sub-optimal, or the agent is required to adapt to new tasks, the multi-game decision transformers need to finetune the model parameters while Gato is required to get prompted with expert demonstrations.

Meta RL. IDT falls into the category of methods of learning to learn, which is also known as meta-learning. More precisely, recent in-context RL methods can be categorized as in-context meta-RL methods. The general idea of learning self-improvement has a long history in RL but is limited to hyper-parameters in the early stages (Ishii et al., 2002). In-context meta-RL methods (Wang et al., 2016; Duan et al., 2016) are commonly trained in the online setting by maximizing multi-episodic value functions with memory-based architectures through environment interactions. Another online meta-RL attempts to find good network parameter initializations and then quickly adapt through additional gradient updates (Finn et al., 2017; Nichol et al., 2018). More recently, meta-RL has seen substantial breakthroughs, from performance gains on popular benchmarks to offline settings, such as Bayesian RL (Dorfman et al., 2021) and optimization-based meta-RL (Mitchell et al., 2021). Considering the difficulty of a completely offline setting, recent work has explored hybrid offline-online settings (Zahavy et al., 2020; Pong et al., 2022). IDT is similar to the hybrid offline-online setting, but the online phase does not involve gradient updates.

In-Context RL. In-context RL is the one that addresses tasks by providing prompts or demonstrations (Chen et al., 2021; Janner et al., 2021; Huang et al., 2024a;b). By training agents at a large scale, transformer-based policies usually have the ability to learn in context (Lee et al., 2022; Reed et al., 2022). The learning process is performed entirely in context and does not involve parameter updates of neural networks. In this work, we consider incremental in-context RL that involves learning from one's own behaviors through a trial-and-error manner. Laskin et al. (2023) proposed Algorithm Distillation (AD) that automatically improves its performance in a trial-and-error manner by providing multiple historical trajectories. Subsequently, Lee et al. (2023b) proposed a Decision-Pretrained Transformer, which trains the agent to find optimal behaviors faster by only predicting the optimal trajectory. More recently, Hao Liu (2023) further demonstrated that across-episodic contexts encourage large transformer models' emerging trial-and-error behaviors. However, these methods focus on the smallest action level, which causes across-episodic contexts to induce too-long sequences and suffer from huge computational costs. In contrast, IDT explores the trial-and-error ability of high-level decisions, which can significantly shorten the length of across-episodic contexts and, therefore, alleviate the computational costs arising from the self-attention mechanism.

3. Preliminaries

Partially Observable Markov Decision Process. We consider learning problems in the context of Partially Observable Markov Decision Processes (POMDP) represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, \mathcal{R})$. The POMDP tuple consist of states $s \in \mathcal{S}$, observations $o \in \mathcal{O}$, actions $a \in \mathcal{A}$, rewards $r \in \mathcal{R}$, and transition probability function $P(s_{t+1}|s_t, a_t)$, where t is an integer denoting the timestep. In environments described by a POMDP, at each timestep t the agent receives the observation o_t, selects an action $a_t \sim \pi(\cdot|o_t)$ from its policy, and then receives the next observation o_{t+1}. A trajectory is a sequence of observations, actions, and rewards and is denoted by $\tau = (o_0, a_0, r_0, \ldots, o_T, a_T, r_T)$. The return of a trajectory at timestep t, $R_t = \sum_{t'=t}^{T} r'_t$, is calculated as the sum of future rewards from that timestep. In addition, a completion token d_t, a binary identifier, is used to indicate whether a trajectory ends at time t.

Hierarchical Reinforcement Learning. RL algorithms aim to maximize the expected return $\mathrm{E}[\sum_{t=0}^{T} r_t]$ throughout an agent's lifetime or training episodes. In long-horizon tasks, standard RL methods suffer from poor performance due to the exponentially growing exploration space. Hierarchical RL decomposes the long-horizon task into subproblems or subtasks such that a high-level policy learns to perform the task by choosing optimal subtasks as the high-level decisions (Pateria et al., 2021). High-level decisions can be designed as discrete or continuous forms. The discrete form can select multiple independent low-level policy models (Bacon et al., 2017), while the continuous form usually serves as additional conditions to control a general low-level policy model (Nachum et al., 2018). Since the transformer-based policy is a conditional generative model, it is naturally adapted to high-level decisions in the continuous form, such as the return-to-go condition in the decision transformer (Chen et al., 2021). In this work, we use a vector $\mathbf{z}$ to represent high-level decisions and assume that it is sampled from a multivariate Gaussian distribution.

Transformers. The Transformer (Vaswani et al., 2017) architecture consists of multiple layers of self-attention operation and MLP. The self-attention begins by projecting input data X with three separate matrices onto D-dimensional vectors called queries Q, keys K, and values V. These vectors are then passed through the attention function:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(QK^T/\sqrt{D})V. \qquad (1)$$

The QK^T term computes an inner product between two projections of the input data X. The inner product is then normalized and projected back to a D-dimensional vector with the scaling term V. Transformers utilize self-attention as a core part of the architecture to process sequential data (Devlin et al., 2018; Brown et al., 2020a). In this work, we use GPT (Radford et al., 2018) architecture that modifies the transformer with a causal self-attention mask to focus on the previous tokens in the sequence ($j \in [1, i]$), enabling us to do autoregressive generation at test time.

4. Method

In this section, we present IDT, which models a high-level trial-and-error process through a hierarchical chain of experience, as summarized in Figure 2.

4.1. Chain of Experience

The key factors that influence our modeling on how to represent trajectories are (1) the ability of transformers to uncover meaningful patterns from multiple trajectories and (2) the

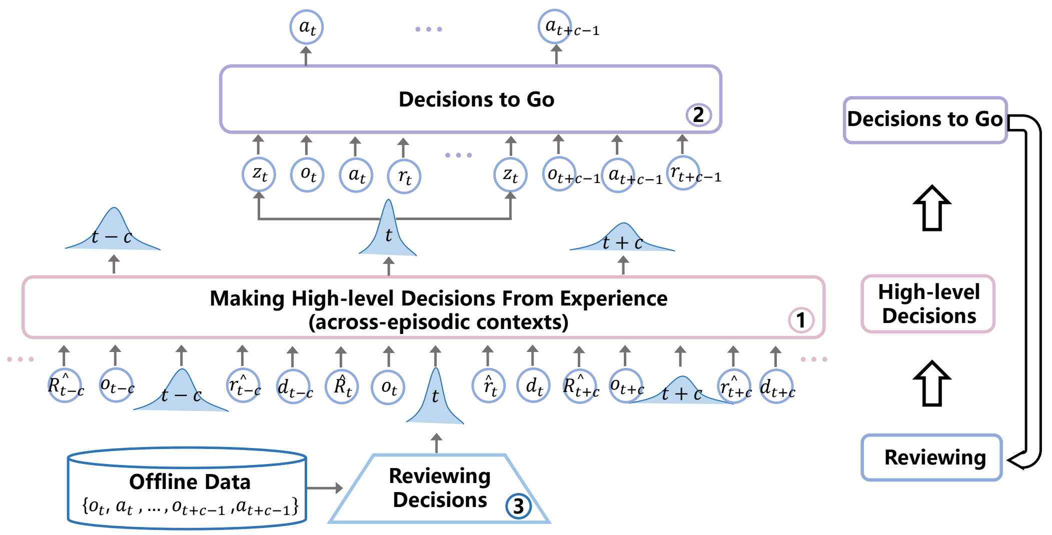
a_t
a_{t+c-1}
Decisions to Go
2
z_t o_t a_t r_t z_t o_{t+c-1} a_{t+c-1} r_{t+c-1}
$t-c$
t
$t+c$
Making High-level Decisions From Experience (across-episodic contexts)
1
$\hat{R}_{t-c}$ o_{t-c} $t-c$ $\hat{r}_{t-c}$ d_{t-c} $\hat{R}_t$ o_t t $\hat{r}_t$ d_t $\hat{R}_{t+c}$ o_{t+c} $t+c$ $\hat{r}_{t+c}$ d_{t+c}
Offline Data
$\{o_t, a_t, \dots, o_{t+c-1}, a_{t+c-1}\}$
Reviewing Decisions
3
Decisions to Go
High-level Decisions
Reviewing

Figure 2. The architecture of IDT is designed into three modules to simulate the high-level trial-and-error process. First, the (1) Making Decisions module predicts a high-level decision by providing across-episodic contexts, where across-episodic contexts contain multiple trajectories arranged in ascending order of the total rewards. Then, the (2) Decisions to Go module predicts actions for c steps conditioned on the predicted high-level decision. Finally, the (3) Reviewing Decisions module reviews the executed actions to serve as an experience for the next cycle. Note that the Reviewing Decisions encodes the true label of high-level decisions from offline data at training while encodes from the executed actions at testing.

capacity to improve itself conditioned on experience. The basic elements of trajectories are observations o, actions a, rewards r, and completion token d. As modeling rewards is a nontrivial task, we aim to have the model generate actions based on the target returns $\hat{R}_0$ (Chen et al., 2021), which can be updated using rewards $\hat{R}_t = \hat{R}_0 - \sum_{j=0}^{t} r_j$. Therefore, the following trajectory representation is amenable to autoregressive training and generation:

$$\tau = (\hat{R}_0, o_0, a_0, r_0, d_0, \dots, \hat{R}_T, o_T, a_T, r_T, d_T). \quad (2)$$

To facilitate the model to achieve target return, we construct across-episodic contexts that consist of multiple trajectories for self-improvement during test time (Hao Liu, 2023). This idea arises from the approach called chain of hindsight (Liu et al., 2023), which trains language models from human feedback by conditioning on positive indicators and negative-rated examples to predict corresponding positive-rated examples. In the RL tasks, the positive indicator is the target return, and previous trajectories serve as negative-rated examples to predict the trajectory with higher returns.

Specifically, the chain of experience is represented by n trajectories: $s = (\tau^1, \tau^2, \dots, \tau^n)$ where

$$\tau^i = (\hat{R}^i_0, o^i_0, a^i_0, r^i_0, d^i_0, \dots, \hat{R}^i_T, o^i_T, a^i_T, r^i_T, d^i_T). \quad (3)$$

The trajectories are ascending sorted according to their total rewards, i.e., $\sum_{t=0}^{T} r^1_t \leq \sum_{t=0}^{T} r^2_t \leq \cdots \leq \sum_{t=0}^{T} r^n_t$. For all n trajectories, the initial target return $\hat{R}^i_0$ equals the max total reward, i.e., the last trajectory $\hat{R}^n_0 = \sum_{t=0}^{T} r^n_t$.

4.2. Hierarchical Chain of Experience

After building across-episodic contexts based on the chain of experience, in-context RL can automatically improve its performance at evaluation time by rolling trajectories in a trial-and-error manner. However, this suffers substantial computational costs when the horizon of tasks increases.

Since total rewards are obtained at the end of episodes, it is more difficult to evaluate and improve a policy model in long-horizon tasks. In traditional RL methods, an effective solution is to decompose complex tasks into several subproblems by incorporating hierarchical structures (Nachum et al., 2018). The high-level policy only needs to generate a signal once to control the low-level policy to generate multi-step actions. This allows (1) the high-level policy to receive feedback faster, as if working on a short-horizon task, and (2) the low-level policy only needs to consider how to better implement the sub-tasks generated by the high-level decision. Although the hierarchical structure can be optimized end-to-end by reward signals, the trial-and-error process in in-context RL is more complicated.

As psychologist Edward Thorndike mentioned, the trial-and-error process includes two parts (Sutton & Barto, 2018), memory and search. The high-level decision plays an important role that is closely connected with memory and search. A high-level decision is generated from the search process and directly affects the quality of low-level executed actions. Subsequently, it also serves as the memory for future searches. Therefore, we designed three modules to realize a high-level trial-and-error process: Making Decisions, Decisions to Go, and Reviewing Decisions.

Making Decisions. The purpose of the Making Decisions module is to generate high-level decisions autoregressively, where the high-level decision is represented by a vector $\mathbf{z}$ sampled from a multivariate Gaussian distribution. As the quality of $\mathbf{z}$ directly relates to low-level actions a, a better high-level decision $\mathbf{z}$ is critical for inducing better low-level actions a. Therefore, we reconstruct across-episodic contexts represented as a high-level chain of experience $s_h = (\tau_h^1, \tau_h^2, \ldots, \tau_h^n)$. Each τ_h^i is denoted as:

$$\begin{aligned}\tau_h^i = (&\hat{R}_0^i, o_0^i, \mathbf{z}_0^i, \hat{r}_0^i, d_0^i, \hat{R}_c^i, o_c^i, \mathbf{z}_c^i, \hat{r}_c^i, d_c^i, \\ &\ldots, \hat{R}_{kc}^i, o_{kc}^i, \mathbf{z}_{kc}^i, \hat{r}_{kc}^i, d_{kc}^i),\end{aligned} \tag{4}$$

where each high-level decision $\mathbf{z}$ is generated every c steps, $T - c \leq kc \leq T$, and $\hat{r}_c^i = \sum_{t=c}^{2c-1} r_t^i$ is the sum of c steps rewards. By comparing with Equation (3), the high-level chain of experience can considerably shorten the length of contexts and, therefore, significantly alleviate the computational complexity of the self-attention mechanism.

Decisions to Go. Based on high-level decisions, the Decisions to Go module is designed to generate low-level actions that can interact with environments. Since the transformer-based policy is a conditional generative model, we can build a low-level context that contains high-level decisions to control the low-level actions. The low-level context is represented as:

$$\begin{aligned}\tau_l^{i,j} =&(\mathbf{z}_j^i, o_j^i, a_j^i, r_j^i, \mathbf{z}_j^i, o_{j+1}^i, a_{j+1}^i, r_{j+1}^i, \\ &\ldots, \mathbf{z}_j^i, o_{j+c-1}^i, a_{j+c-1}^i, r_{j+c-1}^i),\end{aligned} \tag{5}$$

where each $\tau_l^{i,j}$ starts from the generation step $j \in \{0, c, \ldots, kc\}$ of high-level decisions and completes c steps low-level actions in the trajectory τ^i. In particular, we introduce the reparameterization trick (Jang et al., 2016) into high-level decisions to ensure backpropagation through the Decisions to Go module to the Making Decisions module.

Reviewing Decisions. The autoregressive training of the conditional generation model is achieved by predicting each token in the sequence. For example, when the transformer model is trained to generate a_t, we need to provide the action label at time t and condition it on the historical actions $\{a_0, \ldots, a_{t-1}\}$. However, the supervisory signals about high-level decisions $\mathbf{z}$ are not directly observable from the sequence, as shown in Equation (4). For instance, when the transformer model is trained to generate $\mathbf{z}_j$ ($j \in \{0, c, \ldots, kc\}$), we have neither the true label of $\mathbf{z}_j$ nor the previous high-level decisions $\{\mathbf{z}_0, \mathbf{z}_c, \ldots, \mathbf{z}_{j-c}\}$.

To this end, we replace the true label of $\mathbf{z}_j$ with the gradients from the Decisions to Go module trained to generate the following c steps of actions $\{a_j, a_{j+1}, \ldots, a_{j+c-1}\}$. For the previous high-level decisions $\{\mathbf{z}_0, \mathbf{z}_c, \ldots, \mathbf{z}_{j-c}\}$, we introduce the Reviewing Decisions module to encode the label from low-level actions. As the high-level decisions induce low-level actions, the low-level actions should be able to infer high-level decisions inversely. Specifically, to infer a previous high-level decision $\mathbf{z}_t \in \{\mathbf{z}_0, \mathbf{z}_c, \ldots, \mathbf{z}_{j-c}\}$, we first utilize the self-attention operation to aggregate the information of a_{t+c-1} and $\{o_t, a_t, \ldots, o_{t+c-1}, a_{t+c-1}\}$. Then, we apply a linear layer to encode $\mathbf{z}_t$ from the aggregated information. Note that the Reviewing Decision module is not required to perform autoregressive generation, so any sequence model, such as LSTM, can replace it.

By combining the above three modules, IDT can automatically improve its performance at evaluation time by rolling trajectories in a high-level trial-and-error manner. We now introduce the implementation details of IDT, including architecture, training, and testing.

4.3. Implementation of IDT

Architecture. We feed n trajectories into the Making Decisions module, which results in $5 \times n \times T/c$ tokens, with one token for each of the five modalities: desired target return, observation, high-level decision, reward, and completion. In the Decisions to Go module, we feed $4 \times c$ tokens, with one token for each of the four modalities: high-level decision, observation, action, and rewards. In the Reviewing Decisions module, we feed $2 \times c$ tokens, with one token for each of the two modalities: observation and action. To create the token embeddings, we train a linear layer for each modality, which transforms the raw inputs into the desired embedding dimension, followed by layer normalization (Ba et al., 2016). Finally, the tokens are processed by a GPT model that predicts future high- and low-level action tokens through autoregressive modeling.

Training and Testing. During training, we are given a dataset of offline trajectories, where the trajectories can be suboptimal. In each iteration, we sample minibatches of trajectories from the dataset. The Reviewing Decisions module first encodes each true high-level decision $\mathbf{z}$ from the minibatch every c steps. Then, the Making Decisions module predicts the high-level decision $\mathbf{z}_t$ given the input token o_t and past trajectories. Finally, the Decisions to Go module autoregressively predicts c steps of low-level actions $\{a_t, \ldots, a_{t+c-1}\}$ given $\mathbf{z}_t$ and $\{o_t, \ldots, o_{t+c-1}\}$. The low-level actions are evaluated with either cross-entropy loss or mean-squared error, depending on whether the actions are discrete or continuous. The losses from each timestep are averaged and updated in all three modules end-to-end. At test time, we roll out the IDT with multiple trajectories and report the largest return among trajectories. Following the configuration from related works Hao Liu (2023);

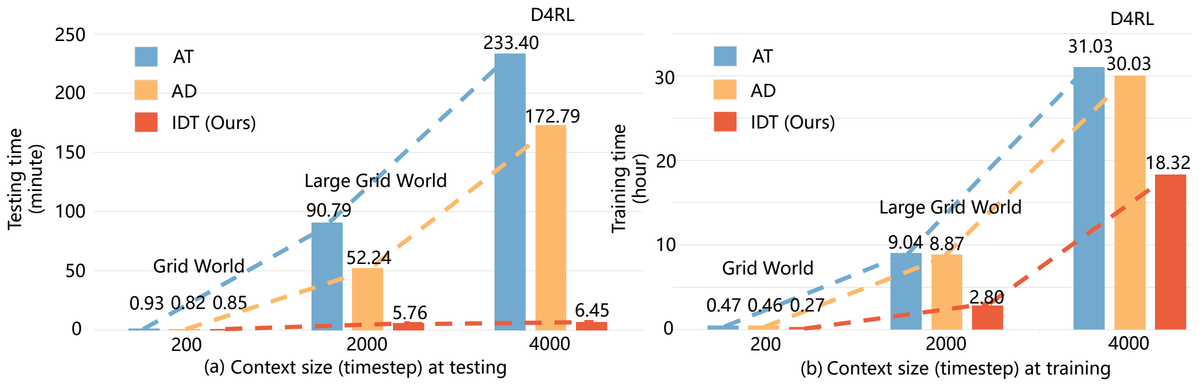

Figure 3. Results for (a) testing and (b) training times. We report the training time per 10k gradient updates, the testing time for 50 episodes over Grid World, and 10 episodes over D4RL. Note that we use the number of steps to measure the context size here. The number of tokens per step may vary depending on the algorithm. Each step in AD contains 4 tokens: observation, action, reward, and completion. IDT's Making Decisions module and AT have an extra return-to-go token. As the task length increases, the context length is forced to grow exponentially, resulting in a square increase in computational costs. In contrast, IDT reconstructs the sequence to consist of high-level decisions. Therefore, the context is smaller than one episode length, significantly reducing computational costs.

Laskin et al. (2023), we set a context size across $n = 4$ episodes. Note that the task horizons T used in this work range from 20 steps to 1000 steps, and the maximum context size reaches 20000 tokens. The pseudocode for IDT is summarized in Appendix A. Source code and more hyperparameters are described in Appendix B.

5. Experiments

Dataset: Grid World. In this section, we first consider the discrete control environments from the Grid World (Lee et al., 2022), which is a commonly used benchmark for recent in-context RL methods. The environments support many tasks that cannot be solved through zero-shot generalization after pre-training because these tasks cannot be inferred easily from the observation. The episode of each task is short enough to train a transformer-based policy with across-episodic contexts feasibly. Specifically, we consider the four evaluation environments: Darkroom, Darkroom Hard, Darkroom Dynamic, and Dark Key-to-Door.

The evaluation environments provide a 2D discrete POMDP where an agent spawns in a room and must find a goal location. The agent only observes its own (x, y) coordinates but does not know the goal location, which is required to deduce it from the rewards received. The room dimensions are 9×9 with the agent's possible actions, including moving one step either left, right, up, down, or staying idle. In Darkroom, an episode lasts 20 steps, and the agent can obtain a reward ($r = 1$) each time the goal is achieved. The Darkroom Hard and Darkroom Dynamic are two variants of Darkroom. In the Darkroom Hard, agents only obtain a reward when the goal is achieved first. In the Darkroom Dynamic, the goal is fixed to a corner, but the action space is randomly permuted. In the Dark Key-to-Door, the length of an episode is 50, where the agent is required to locate an invisible key to receive a one-time reward first and then identify an invisible door to obtain another one-time reward.

In addition, we create a variant of Large Darkroom, Large Darkroom Hard, Large Darkroom Dynamic, and Large Darkroom Key-to-Door, where the coordinate space of each environment is expanded to 40×40, and the episode length is expanded 10 times. The dataset is collected from learning histories that are generated by training gradient-based RL algorithms, such as Deep Q-Network (Mnih et al., 2013). For each environment, we randomly create 60 tasks from the coordinate space and collect data for 1 million timesteps.

Dateset: D4RL. D4RL (Fu et al., 2020) is a commonly used offline RL benchmark, including continuous control tasks. The different dataset settings are described below.

- Medium: 1 million timesteps generated by a "medium" policy that performs approximately one-third as well as an expert policy.
- Medium-Replay: 1 million timesteps collected from the replay buffer of an agent trained to the performance of a "medium" policy.
- Medium-Expert: It consists of 1 million timesteps generated by the "medium" policy and another 1 million timesteps generated by the expert policy.

The dataset is collected from Mujoco environments, including HalfCheetah, Hopper, and Walker. The episode length in D4RL is 1000, which is far more than that of Grid World.

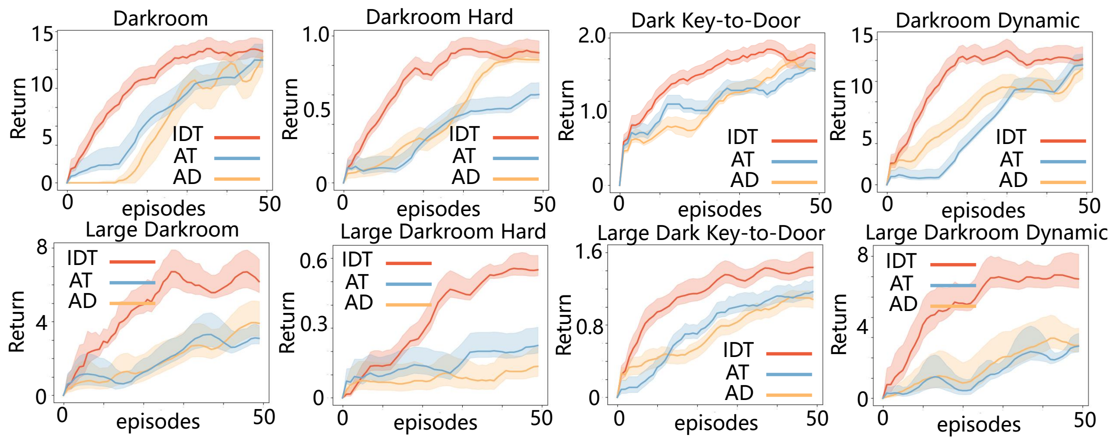

Figure 4. Results for Grid World. An agent is expected to solve a new task by interacting with the environments for 50 episodes without online model updates. Based on high-level decisions, our method outperforms both AT and AD, which rely on across-episodic contexts with the smallest actions. In particular, IDT has significant advantages in handling long-horizon tasks.

Therefore, current in-context RL methods require huge computational costs in D4RL, even though it is a commonly used baseline for conventional RL algorithms.

Baselines. In this section, we investigate the performance and efficiency of IDT relative to in-context RL, dedicated offline RL, and imitation learning algorithms. Our baselines can be categorized as follows:

- In-context RL: These methods use the transformer to model trajectory sequences and predict actions autoregressively. We compare with recent methods, Agentic Transformer (AT) (Hao Liu, 2023) and Algorithm Distillation (AD) (Laskin et al., 2023), which proposed across-episodic contexts with the smallest actions.
- Temporal-difference learning: Most temporal-difference (TD) learning methods use an action space constraint or value pessimism and will serve as faithful comparisons to IDT, representing standard RL methods. We consider state-of-the-art TD3+BC (Fujimoto & Gu, 2021) that is demonstrated to be effective on D4RL.
- Imitation learning: Imitation learning methods similarly utilize supervised losses for training, such as Behavior Cloning (BC) (Torabi et al., 2018) and Decision Transformer (DT) (Chen et al., 2021). Following AT, we compare with BC-10%, which is shown to be competitive with state-of-the-art on D4RL. DT also uses a transformer to predict actions autoregressively but is limited to a single episode context.

For all comparison methods, we adhere closely to the original hyper-parameter settings. To evaluate IDT and other in-context RL algorithms, we roll out 10 episodes in D4RL and 50 episodes in Grid World. For each result, we report mean and standard error across 5 random seeds.

5.1. Evaluation of Computing Costs

An important property of in-context RL is that it can improve itself without expensive gradient updates. However, the computational costs of forward propagation are hidden in short-horizon tasks. Therefore, we reported the training time per 10k gradient updates, the evaluation time for 50 episodes over Grid World, and 10 episodes over D4RL. As shown in Figure 3, our IDT has efficient training and significantly reduces the testing time compared to the baselines, approximately **36**$\times$ times faster in D4RL and **27**$\times$ times faster in large Grid World. More detailed results for each task are described in Appendix C.

As the task length increases, the evaluation time of AT and AD grows quadratically. This is because the across-episodic contexts multiply the sequence length, leading to intolerable computational costs in the self-attention mechanism. The AT algorithm requires four episodes for trial-and-error, where each episode reaches 1000 steps in D4RL, and each step contains 5 tokens. Therefore, each step of AT generation requires scanning 20k tokens. Since the AD algorithm reduces a return-to-go token at each step, the training and testing time are both less than AT. In contrast, IDT reconstructs the sequence to consist of high-level decisions, and thus, the context is smaller than one episode length. As a result, IDT is significantly lower than baselines at both training and testing times.

5.2. Grid World Results

To evaluate IDT's self-improvement capabilities in unseen tasks, we compared recent in-context RL methods in the

Table 1. Results for D4RL datasets. IDT outperforms both in-context RL (AT and AD) and supervised learning (BC) and performs competitively with conventional RL algorithms (TD3+BC and TD3) on almost all tasks.

Dataset	Environment	BC-10%	TD3+BC	TD3	DT	AT	AD	Ours
Medium-Expert	HalfCheetah	94.11	**96.59**	87.60	93.40	95.81± 0.25	94.21 ± 0.46	96.12± 0.18
Medium-Expert	Hopper	113.13	113.22	98.41	111.18	115.92± 1.26	108.32 ± 0.95	**118.39± 0.75**
Medium-Expert	Walker	109.90	112.21	100.52	108.71	114.87± 0.56	111.36 ± 0.46	**118.51± 0.48**
Medium	HalfCheetah	43.90	**48.93**	34.60	42.73	45.12± 0.34	42.28 ± 1.18	45.51± 0.26
Medium	Hopper	73.84	70.44	56.98	69.42	70.45± 0.45	72.58 ± 0.54	**83.24± 0.33**
Medium	Walker	82.05	86.91	70.95	74.70	88.71± 0.55	85.96 ± 0.46	**88.94± 0.61**
Medium-Replay	HalfCheetah	42.27	45.84	38.81	40.31	**46.86± 0.33**	41.28 ± 0.21	45.58± 0.36
Medium-Replay	Hopper	90.57	98.12	78.90	88.74	96.85± 0.41	91.32 ± 0.66	**98.59± 0.26**
Medium-Replay	Walker	76.09	91.17	65.94	68.22	92.32± 1.21	89.21 ± 1.42	**96.22± 1.06**
Total Average		80.65± 1.34	84.83± 1.10	70.28± 1.20	77.49± 1.45	85.21± 1.12	82.84± 0.70	**87.90± 1.06**

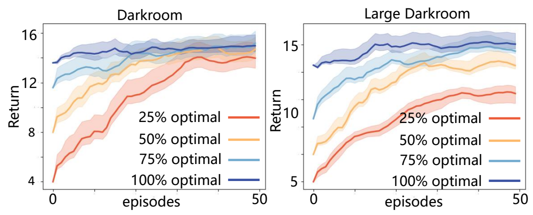

Figure 5. Results for IDT conditioned on partial demonstrations. IDT can accelerate self-improvement through the Review Decisions module to encode external data prompts.

Grid World environments. The agent is required to solve an unseen task by interacting with the environments for 50 episodes without online model updates. As shown in Figure 4, IDT achieves state-of-the-art performance in a wide range of tasks.

In large variant tasks, IDT significantly surpasses the baselines in both efficiency and performance. However, neither AT nor AD showed obvious self-improvement trends, especially in Large Darkroom Hard. This is because the Large Darkroom Hard is a task with sparse rewards, which makes it difficult for AT and AD to capture the goal position in long sequences. In contrast, IDT explores tasks in a high-level trial-and-error manner, making receiving positive feedback on rewards easier. Overall, IDT demonstrated that a high-level trial-and-error manner is feasible rather than limited to the smallest actions.

5.3. D4RL Results

In addition to short-horizon tasks specific to in-context RL methods, we also test the performance of IDT on the D4RL dataset, which is commonly used in conventional RL methods. Based on Fu et al. (2020), the results on D4RL are normalized so that 100 denotes an expert policy. Baseline numbers are reported by the AT paper and from the D4RL paper. As shown in Table 1, IDT outperforms baselines in a majority of the tasks and is competitive with the state-of-the-art in the remaining tasks.

In the TD learning and imitation learning categories, TD3+BC is generally the most remarkable algorithm. Compared with them, the superior performance of IDT demonstrates the advantages of using high-level trial-and-error.

5.4. Case Study on the Reviewing Decisions Module

A notable ability of transformer-based policy is to address tasks by providing demonstration prompts. Although in-context RL can improve itself without relying on demonstrations, external prompts can speed up the process. Therefore, we want to investigate whether IDT can benefit from this setting. To answer this question, we design a ϵ-greedy policy to collect external data in Darkroom and Large Darkroom tasks, ranging from nearly random to optimal.

As shown in Figure 5, IDT improves each policy in context until it is near-optimal. Notably, the more optimal the input policy, the faster IDT improves it until it is optimal. Despite high-level decisions that cannot be directly observed from the external demonstrations, IDT can extract experts' intentions through the Reviewing Decisions module. In particular, we also perform parameter sensitivity analyses on high-level decision frequency (c) and context size (n episodes), as shown in Appendix C.

6. Conclusion

In this work, we propose an efficient in-context RL method IDT that treats RL tasks as an across-episodic sequence problem and can improve itself at test time. The idea of human multi-level decision-making inspires IDT and introduces high-level decisions into the sequence prediction process. Unlike current in-context RL methods limited to short-horizon tasks, IDT is also good at standard RL benchmarks, which typically have longer task horizons. On the

Grid World and D4RL benchmarks, we show that IDT can outperform baselines in both efficiency and performance.

Impact Statement

In terms of the potential broader impact, our work provides a new idea of incorporating high-level decisions in the design of context, which may promote the development of the in-context RL community. Besides, we do not see any negative ethical and societal impacts of our work while using our method in practice.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2021ZD0112500; the National Natural Science Foundation of China under Grant Nos. U22A2098, U19A2065, 62172185, 61976102, 62206105 and 62202200; the International Cooperation Project of Jilin Province under Grant Nos.20220402009GH, U2341229.

References

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

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*, 47:253–279, 2013.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020a.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020b.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

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. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Hao Liu, P. A. Emergent agentic transformer from chain of hindsight experience. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2023)*, 2023.

Hu, J., Sun, Y., Chen, H., Huang, S., Chang, Y., Sun, L., et al. Distributional reward estimation for effective multi-agent deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:12619–12632, 2022.

Hu, J., Sun, Y., Huang, S., Guo, S., Chen, H., Shen, L., Sun, L., Chang, Y., and Tao, D. Instructed diffuser with temporal condition guidance for offline reinforcement learning. *arXiv preprint arXiv:2306.04875*, 2023.

Huang, S., Hu, J., Yang, Z., Yang, L., Luo, T., Chen, H., Sun, L., and Yang, B. Decision mamba: Reinforcement learning via hybrid selective sequence modeling. *arXiv preprint arXiv:2406.00079*, 2024a.

Huang, S., Sun, Y., Hu, J., Guo, S., Chen, H., Chang, Y., Sun, L., and Yang, B. Learning generalizable agents via saliency-guided features decorrelation. *Advances in Neural Information Processing Systems*, 36, 2024b.

Ishii, S., Yoshida, W., and Yoshimoto, J. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687, 2002.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S. S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations*, 2023.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023a.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023b.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

Liu, H., Sferrazza, C., and Abbeel, P. Chain of hindsight aligns language models with feedback. *arXiv preprint arXiv:2302.02676*, 3, 2023.

Mitchell, E., Rafailov, R., Peng, X. B., Levine, S., and Finn, C. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Pong, V. H., Nair, A. V., Smith, L. M., Huang, C., and Levine, S. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pp. 17811–17829. PMLR, 2022.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999a.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999b.

Thorndike, E. L. The law of effect. *The American journal of psychology*, 39(1/4):212–222, 1927.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Zahavy, T., Xu, Z., Veeriah, V., Hessel, M., Oh, J., van Hasselt, H. P., Silver, D., and Singh, S. A self-tuning actor-critic algorithm. *Advances in neural information processing systems*, 33:20913–20924, 2020.

Appendix of paper "In-Context Decision Transformer: Reinforcement Learning via Hierarchical Chain-of-Thought"

A. Pseudocode of In-context Decision Transformer

Algorithm 1 In-context Decision Transformer.

```
1: Input: A dataset of Trajectories, Max Iterations $M$ as training phase, Max episodes $m$ at testing phase, A number of trajectories $n$ in hierarchical chain of experience, A number of steps of low-level actions $c$ for one high-level decision
2: Output: The generated low-level actions
3: // Training
4: for $i = 1$ to $M$ do
5:     Randomly sample $n$ episodes from dataset $s = (\tau^1, \tau^2, \dots, \tau^n)$
6:     Sort $n$ episodes ascending according to their returns $\sum_{t=0}^{T} r_t^1 \leq \sum_{t=0}^{T} r_t^2 \leq \cdots \leq \sum_{t=0}^{T} r_t^n$
7:     Compute returns-to-go $\hat{R}_t = \hat{R}_0 - \sum_{j=0}^{t} r_j$ for all steps for each episode, where $\hat{R}_0 = \sum_{t=0}^{T} r_t^n$
8:     The Reviewing Decisions module encodes a high-level decisions $\mathbf{z}$ every $c$ steps
9:     Concatenate $n$ episodes as a high-level sequence $s_h = (\tau_h^1, \tau_h^2, \dots, \tau_h^n)$ based on Equation (4)
10:    Build a low-level sequence every $c$ steps based on Equation (5)
11:    The Making Decisions module predicts the next high-level decision tokens, and then the Decision to Go module predicts the next $c$ steps low-level action tokens for each predicted high-level decision token
12:    Train the Reviewing Decisions, Making Decisions, and Decision to Go modules based on the loss of predicted low-level actions end-to-end
13: end for
14: // Testing
15: for $i = 1$ to $m$ do
16:     Start a new episode $i$ and reset the timestep $t = 0$
17:     while $t \leq T$ do
18:         The Making Decisions model generates next high-level decision token $\mathbf{z}_t^i$ based on the across-episodic context $(\tau_h^{i-3}, \tau_h^{i-2}, \tau_h^{i-1}, \dots, \hat{R}_t^i, o_t^i)$, where $\tau_h^i$ is expressed as Equation (4)
19:         for $k = 0$ to $c - 1$ do
20:             The Decisions to Go model generates next low-level action $a_{t+k}^i$ based on the previous context $(\mathbf{z}_t^i, o_t^i, a_t^i, r_t^i, \dots, \mathbf{z}_t^i, o_{t+k}^i)$
21:         end for
22:         The Review Decisions model encode the executed decision $\mathbf{z}_t^i$ from the $c$ steps $(o_t^i, a_t^i, \dots, o_{t+c-1}^i, a_{t+c-1}^i)$
23:         Compute the sum of $c$ steps rewards $\hat{r}_t^i$ and the next returns-to-go $\hat{R}_{t+c}^i$
24:         Receive the next observation $o_{t+c}^i$
25:         Update the across-episodic context $((\tau_h^{i-3}, \tau_h^{i-2}, \tau_h^{i-1}, \dots, \hat{R}_t^i, o_t^i, \mathbf{z}_t^i, \hat{r}_t^i, d_t^i, \hat{R}_{t+c}^i, o_{t+c}^i)$
26:         Update time step $t = t + c$
27:     end while
28: end for
```

In Algorithm 1, we introduce the training and testing process of IDT. At each iteration, we first construct a sequence consisting of high-level decisions, as described in lines 5-9. Importantly, high-level decisions in the dataset are encoded by the Reviewing Decisions module (line 8). In addition, each high-level decision will correspond to a short sequence of c steps low-level actions, as described in line 10. Based on the constructed sequences, the Making Decisions and Decisions to Go modules predict high-level decisions and low-level actions, respectively (line 11). Finally, the low-level actions are evaluated with either cross-entropy loss or mean-squared error, depending on whether the actions are discrete or continuous. The losses from each time step are averaged and updated in all three modules end-to-end, as described in line 12.

During testing, IDT needs to generate low-level actions autoregressively and interact with the environment m episodes. At step t of episode i (line 18), the Making Decisions module first generates a high-level decision token $\mathbf{z}_t^i$ conditioned on the across-episodic context $(\tau_h^{i-3}, \tau_h^{i-2}, \tau_h^{i-1}, \dots, \hat{R}_t^i, o_t^i)$, where τ_h^i is expressed as Equation (4). Then, the Decision to Go will generate the following c steps low-level actions $(a_t, \dots, a_{t+c-1})$ autoregressively, as described in lines 19-21. Unlike training, the Reviewing Decisions module encodes the executed decision from the actions generated by the Decision to Go

Table 2. Hyperparameters of IDT.

	Hyperparameters	Value
	Number of layers	3
	Number of attention heads	3
	Embedding dimension	128
	Activation function	ReLU
	c steps controlled by one high-level decision	10 D4RL and Large Grid World 5 Grid World
Training	Batch size	64
	Dropout	0.1
	Learning rate	1e-4
	Learning rate decay	Linear warmup for 1e5 steps
	Grad norm clip	0.25
	Weight decay	1e-4
	Number of trajectories to form across-episodic contexts n	4 (Large) Dark Key-to-Door 10 other tasks in Grid World 4 D4RL
Testing	Target return for HalfCheetah	12000
	Target return for Hopper	3600
	Target return for Walker	5000
	Target return for Darkroom	20
	Target return for Darkroom Hard	1
	Target return for Darkroom Dynamic	20
	Target return for Darkroom Key-to-Door	2
	Target return for Large Darkroom	15
	Target return for Large Darkroom Hard	1
	Target return for Large Darkroom Dynamic	15
	Target return for Large Darkroom Key-to-Door	2
	Number of trajectories to form across-episodic contexts n	4 (Large) Dark Key-to-Door 10 other tasks in Grid World 4 D4RL

module. Then, it serves as a condition for generating the next high-level decision $\mathbf{z}^i_{t+c}$, as described in lines 22-26.

B. Experimental Details

Source code is available at here.

Compute. Experiments are carried out on NVIDIA GeForce RTX 3090 GPUs and NVIDIA A10 GPUs. Besides, the CPU type is Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz. Since our memory is not enough to support AT training in D4rl tasks, we refer to the results of the original paper. In contrast, our method has lower memory requirements because it naturally shortens the across-episodic contexts.

Hyperparameters. The default length of across-episodic is four trajectories unless mentioned otherwise. In D4RL and Large Grid World, the Decisions to Go module generates $c = 10$ steps low-level actions while the Making Decisions module generates one high-level decision. In conventional Grid World, we set $c = 5$ because the task is too short. Except for independent parameters and different input and output dimensions, three modules in IDT follow the same architecture. In summary, Table 2 shows the hyperparameters used in our IDT model.

C. Additional Experimental Results

Parameter Sensitive Analysis of c. An important insight of IDT is that one high-level decision can guide c-step low-level actions. We aim to investigate whether the size of c will affect the performance of IDT. Therefore, we tested $c = 1, 5, 10, 15, 20$ in D4RL and Grid World, respectively. As shown in Figure 6, IDT maintains stable performance to

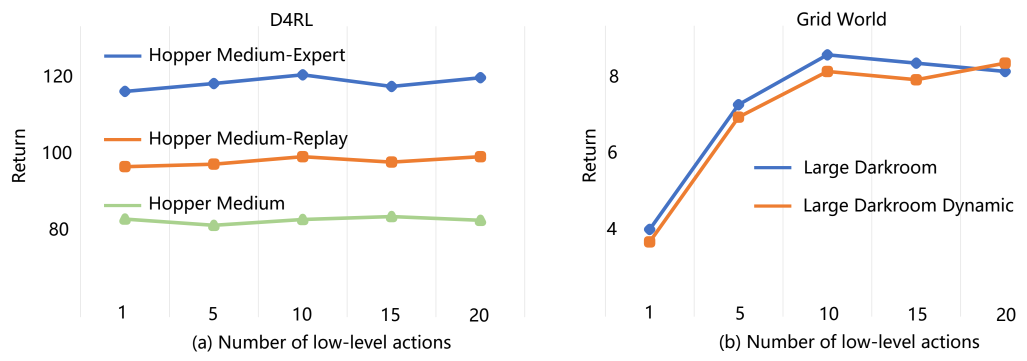

Figure 6. Parameter sensitive analysis of c. (a) IDT maintains stable performance to changes in c in dense reward D4RL tasks. (b) As c increases, it becomes easier for the model to receive positive feedback to discover the target location in sparse reward Grid World tasks.

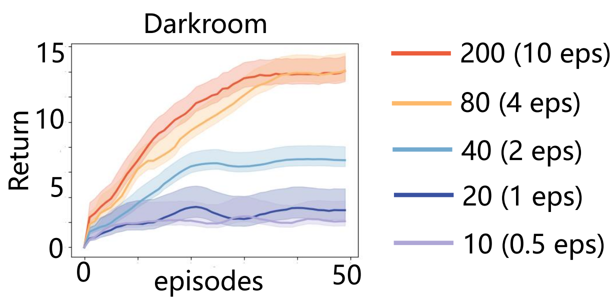

Figure 7. Context size: IDT in Darkroom with different context sizes. IDT emerges with trial-and-error ability once the context size is large enough and across-episodic.

Table 3. Results for training and testing times. We report the training time per 10k gradient updates, the testing time for 50 episodes over Grid World, and 10 episodes over D4RL. As the task length increases, the context length is forced to grow exponentially, resulting in a square increase in computational costs. In contrast, IDT completes trial-and-error on high-level decisions in sizes smaller than one episode length, significantly reducing computational costs.

Context size (step)	Tasks	Training (hour)			Testing (minute)		
		AT	AD	Ours	AT	AD	Ours
200	Darkroom	0.27	0.23	0.21	0.62	0.61	0.65
	Darkroom Hard	0.29	0.28	0.20	0.59	0.56	0.58
	Darkroom Dynamic	0.33	0.31	0.21	0.65	0.62	0.67
	Dark Key-to-Door	1.12	1.01	0.44	1.89	1.50	1.52
2000	Large Darkroom	5.09	4.70	2.49	67.22 (13×)	45.08 (9×)	5.27
	Large Darkroom Hard	6.48	6.69	2.93	66.81 (11×)	44.96 (7×)	6.09
	Large Darkroom Dynamic	5.71	5.84	2.73	62.06 (11×)	42.12 (8×)	5.51
	Large Dark Key-to-Door	18.87	18.23	3.06	167.07 (27×)	76.79 (12×)	6.18
4000	HalfCheetah	36.18	37.10	21.90	234.20 (37×)	173.11 (28×)	6.29
	Walker	32.82	33.77	20.08	233.18 (36×)	172.34 (26×)	6.51
	Hopper	24.08	22.23	12.99	232.82 (35×)	172.92 (26×)	6.56

changes in c in D4RL tasks. In contrast, larger c achieves better performance in Grid World. This is because the Grid World is designed for tasks with sparse rewards where the agent needs to rely on rewards to reason about the target location. As c increases, it becomes easier for the model to receive positive feedback to discover the target location.

What Context Size is Required for IDT? Similar to other in-context RL methods, we also test how context sizes are required for IDT emerging with trial-and-error ability. As shown in Figure 7, multi-episodic contexts of 4 episodes are necessary to learn a near-optimal IDT. When the context size is roughly the length of an episode, IDT begins to emerge with self-improvement. The reason for this is likely that the context is large enough to retrain across-episodic information – e.g., at the start of a new episode, the context will be filled with transitions from most of the previous episode.