Towards Provable Emergence of In-Context Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Typically, a modern reinforcement learning (RL) agent solves a task by updating its neural network parameters to adapt its policy to the task. Recently, it has been observed that some RL agents can solve a wide range of new out-of-distribution tasks without parameter updates after pretraining on some task distribution. When evaluated in a new task, instead of making parameter updates, the pretrained agent conditions its policy on additional input called the context, e.g., the agent's interaction history in the new task. The agent's performance increases as the information in the context increases, with the agent's parameters fixed. This phenomenon is typically called in-context RL (ICRL). The pretrained parameters of the agent network enable the remarkable ICRL phenomenon. However, many ICRL works perform the pretraining with standard RL algorithms. This raises the central question this paper aims to address: Why can the RL pretraining algorithm generate network parameters that enable ICRL? We hypothesize that the parameters capable of ICRL are minimizers of the pretraining loss. This work provides initial support for this hypothesis through a case study. In particular, we prove that when a Transformer is pretrained for policy evaluation, one of the global minimizers of the pretraining loss can enable in-context temporal difference learning.

1 Introduction

Reinforcement learning (RL, Sutton and Barto [2018]) is a powerful machine learning paradigm for solving sequential decision-making problems via trial and error. In deep RL [Mnih et al., 2015, Schulman et al., 2015, 2017], a classic workflow is to first train an agent with an RL algorithm in a task and then deploy it in the same task. Once the task changes, the skills acquired by the agent will be obsolete, and a new agent must typically be trained for the new task by repeating the expensive procedure above. In-context RL (ICRL, Moeini et al. [2025]), first coined by Laskin et al. [2023], has the potential to address this limitation. In short, ICRL enables the agent to learn in the forward pass of their neural network by adding some additional data, called the context, to the network's input. This learning in the forward pass occurs without any parameter updates. Specifically, in ICRL, the RL agent is pretrained on a set of pretraining tasks. After pretraining, the RL agent network's parameters stay fixed, and the agent gets tested on some new test tasks, which can significantly differ from the pretraining tasks. For example, in Laskin et al. [2023], the test task is a bandit problem having opposite optimal arms to the bandit problems in the pretraining tasks. Despite this discrepancy between pretraining and test tasks, one observes that the agent's performance in the test task increases as the context grows, with the agent's parameters frozen. We recall that the context serves as the additional input to the agent network, and a representative example is the agent's historical observations and actions in the new task until the current time step [Laskin et al., 2023]. This performance improvement cannot be due to the hypothesis that the fixed pretrained parameters memorize the pretraining tasks because the test task can drastically differ from the pretraining tasks.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

The only plausible explanation seems to be that the pretrained parameters enable some reinforcement learning process in the forward pass of the network to process the information in the context and then output a good action. This inference-time RL process is called ICRL. Notably, by tasks, we include both policy evaluation and control problems. The former predicts the value function, while the latter searches for a performant policy.

ICRL has a diverse array of pretraining algorithms, which can be divided into supervised pretraining and reinforcement pretraining. The key idea of supervised pretraining is to perform imitation learning to train the agent network to imitate the behaviour of some existing RL algorithms. It is thus not surprising that the pretrained agent network behaves like an RL algorithm in the forward pass. We defer more discussion about supervised pretraining to Section 2. What is surprising is the reinforcement pretraining, where the agent network is some sequence model (e.g., Transformer [Vaswani et al., 2017]) and the pretraining algorithm is merely (a variant of) standard deep RL algorithms [Duan et al., 2016, Wang et al., 2016, Kirsch et al., 2022, Lu et al., 2023, Bauer et al., 2023, Grigsby et al., 2023, 2024, Park et al., 2024, Xu et al., 2024]. Essentially, in reinforcement pretraining, the network is trained to output good action (for control tasks) or to make good value predictions (for policy evaluation tasks). It, however, turns out that the forward pass of the pretrained network behaves like an RL algorithm. We now give a few examples of reinforcement pretraining. Duan et al. [2016] use recurrent neural networks (RNNs) as the agent network and use trust region policy optimization [Schulman et al., 2015] as the pretraining algorithm. The proposed RL^2 algorithm performs impressively in multi-armed bandits, tabular environments, and visually rich domains. A closely related and concurrent work is by Wang et al. [2016], where they train RRNs with A3C [Mnih et al., 2016]. Grigsby et al. [2023] propose Amago that trains a variant of Transformer [Vaswani et al., 2017] with customized REDQ [Chen et al., 2021], which works competitively in challenging robot benchmarks. Wang et al. [2025] train Transformers with temporal difference learning (TD, Sutton [1988]) and find that the pretrained Transformers perform well on unseen policy evaluation tasks. The pretraining algorithm is just a standard RL algorithm in all these examples. It gives rise to two important questions about the learned parameters:

(i) How can the learned parameters enable ICRL in the network's forward pass?

(ii) Why do those parameters emerge after using the standard RL algorithm for pretraining?

Both questions are challenging since answering them requires white-boxing the learned neural network's internals and understanding the dynamics of the pretraining algorithm. To our knowledge, no prior work can answer both questions. In addition, given the diversity of the pretraining algorithms, having a single answer to all pretraining algorithms is unlikely. We present a case study following Wang et al. [2025] in this work.

Wang et al. [2025] answer Question (i) for policy evaluation tasks. In particular, they use TD to pretrain a Transformer for policy evaluation tasks on multiple randomly generated MDPs. After the pretraining converges, they study the converged parameters. Surprisingly, Wang et al. [2025] prove that with the converged parameters, the layer by layer forward pass of their Transformer is equivalent to iteration by iteration updates of TD. In other words, the converged Transformer performs policy evaluation in its forward pass by implementing TD in the forward pass. However, Wang et al. [2025] only partially answer Question (ii). In particular, they show thay the converged parameters belong to an invariant set of the the pretraining algorithm (i.e., TD). By an invariant set, we mean a set of parameters such that once the pretraining algorithm enters, it remains in the set forever (in expectation). For example, if the parameter is in $\mathbb{R}^d$, then a trivial invariant set is just $\mathbb{R}^d$ itself. The invariant set characterized by Wang et al. [2025] is much smaller than $\mathbb{R}^d$ (so it is nontrival and contains valuable information). But Wang et al. [2025] fall short in three aspects. First, their analysis considers only a single-layer Transformer. Second, their invariant set contains more parameters than the converged parameters. Third, whether the pretraining algorithm will always converge to the invariant set remains unclear. This work builds upon Wang et al. [2025] and provides finer answers to Questions (i) and (ii) by making the following three contributions.

1. (Inference Time Convergence) We prove that when a Transformer is parameterized with the converged parameters observed by Wang et al. [2025], the value prediction error made by the Transformer diminishes as the depth of the Transformer grows to infinity.

2. (Global Minimizer of Pretraining Loss) We prove that the converged parameters observed by Wang et al. [2025] is a global minimizer of a few pretraining algorithms, including both TD and Monte Carlo.
3. We empirically verify our theoretical insights with controlled environments mirroring our setup in theory.

2 Related Works

One can categorize the pretraining of ICRL into supervised pretraining and reinforcement pretraining. The key idea of supervised pretraining is imitation learning, or algorithm distillation [Laskin et al., 2023]. In supervised pretraining, a dataset is first constructed by running a few existing known RL algorithms on the pretraining tasks and then collecting all the trajectories generated during the execution of those RL algorithms. Notably, a trajectory starts from the initialization of the RL algorithm and runs until the RL algorithm converges, thus spanning multiple episodes. As a result, the rewards near the beginning of the trajectory are lower, and those near the end are higher. The trajectory thus demonstrates the RL policy's learning progress. Then, a sequence model is used to fit the trajectories in the dataset. Namely, for a trajectory at time t, the input to the sequence model is all the data in the trajectory before time t. The target output of the sequence model is the action taken at time t. The sequence model is trained via an imitation learning loss to output the target action. The sequence model thus mimics the behaviour of RL algorithms. Notable ICRL works with supervised pretraining include Laskin et al. [2023], Liu and Abbeel [2023], Zisman et al. [2023], Shi et al. [2024], Huang et al. [2024a,b], Dai et al. [2024]. Overall, we argue that in supervised pretraining, it is not surprising that the sequence model behaves like an RL algorithm in the forward pass because it was trained to do so. Lin et al. [2023] provide a theoretical analysis of supervised pretraining. In particular, they prove that with supervised pretraining, the pretrained sequence model behaves like the RL algorithms used to generate the dataset indeed. Lin et al. [2023] also provide some parameter construction of Transformers, such that the Transformer with those parameters can indeed implement some RL algorithms in the forward pass, including LinUCB [Chu et al., 2011], Thompson sampling [Russo et al., 2018], and UCB-VI [Azar et al., 2017]. However, the parameter construction in Lin et al. [2023] is overly complicated, and no evidence shows the constructed parameters are learnable. Our work is different from Lin et al. [2023] in that (1) we study reinforcement pretraining, whose dynamics are entirely different from supervised pretraining; (2) the Transformer parameters we study are learnable by practical and standard RL algorithms, which we empirically demonstrate.

In terms of reinforcement pretraining, the closest to our work is Park et al. [2024], which studies ICRL with large language models (LLMs) and proposes a regret-loss for reinforcement pretraining. They prove by construction that the global optimizer of their regret-loss implements the FTRL algorithm [Shalev-Shwartz and Singer, 2007]. While we classify Park et al. [2024] as reinforcement pretraining, it certainly deviates from standard RL algorithms (cf. the reinforcement pretraining algorithms in Section 1). Our work differs from Park et al. [2024] in that (1) the pretraining algorithm is different (so the pretraining dynamics are entirely different); (2) Park et al. [2024] only study single-layer Transformers, while our results apply to multi-layer Transformers; (3) No evidence exists to support that the parameters studied by Park et al. [2024] are learnable empirically, while the Transformer parameters we study are learnable by practical and standard RL algorithms, which we empirically demonstrate.

ICRL falls into the category of black box meta RL. See Beck et al. [2023] for a more comprehensive treatment of meta RL. ICRL is also a special case of in-context learning (ICL, Brown et al. [2020]) if we use ICL to denote any learning paradigm that occurs in the inference time of the neural network. More commonly, ICL exclusively refers to inference-time supervised learning. A line of works studies the provable emergence of in-context supervised learning [Ahn et al., 2024, Zhang et al., 2024, Gatmiry et al., 2024]. The techniques we use in this paper are entirely different from those works because the dynamics of supervised learning and reinforcement learning are completely different.

3 Preliminaries

We use I_n to denote an $n \times n$ identity matrix and $0_{m\times n}$ to denote an all-zero matrix in $\mathbb{R}^{m\times n}$ in this work. Given two vectors x, y, we will use $\langle x, y\rangle$ and $x^\top y$ to denote their inner product interchangeably.

3.1 Policy Evaluation

We consider an infinite-horizon Markov decision process (MDP, Puterman [2014]) with finite state space $\mathcal{S}$ and action space $\mathcal{A}$, a bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a state transition probability function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, an initial distribution $p_0 : \mathcal{S} \to [0, 1]$, and a discount factor $\gamma \in [0, 1)$. An agent implements a policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$. Suppose the agent is at state S_t at time step t, it outputs an action $A_t \sim \pi(\cdot \mid S_t)$, receives a reward $R_{t+1} \doteq r(S_t, A_t)$ and transitions to the next state $S_{t+1} \sim p(\cdot \mid S_t, A_t)$. When the policy π is fixed, the MDP reduces to a Markov reward process (MRP), characterized by the tuple $\langle \mathcal{S}, p, p_0, r, \gamma \rangle$, where $p(s' \mid s) \doteq \sum_{a \in \mathcal{A}} \pi(a \mid s) p(s' \mid s, a)$ and $r(s) \doteq \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a)$. The goal of policy evaluation is estimating the value function $v : \mathcal{S} \to \mathbb{R}$, defined as $v(s) \doteq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s\right]$. Since we will only consider policy evaluation tasks in this work, we will work with MRPs for simplicity. It is common to approximate v with some parametric model $\hat{v}_w$ where $w \in \mathbb{R}^d$ is the weight. One of the simplest function approximators is arguably the linear model. Suppose there exists some feature mapping $\phi : \mathcal{S} \to \mathbb{R}^d$, a linear function approximation of v is then $\hat{v}_w(s) \doteq \phi(s)^\top w \approx v(s)$. Note that p defines a Markov chain. Assuming the chain is ergodic, there exists a unique and well-defined stationary distribution $\mu : \mathcal{S} \to [0, 1]$. Then, a common measure of the approximation error called the mean square value error (MSVE) can be well-defined as $\text{MSVE}(v, v') \doteq \sum_{s \in \mathcal{S}} \mu(s)(v(s) - v'(s))^2$, for $v', v : \mathcal{S} \to \mathbb{R}$. Thus, one typically wishes to find w^* such that $\text{MSVE}(v, \hat{v}_{w^*})$ is minimized.

Monte Carlo (MC, Sutton and Barto [2018]) is a straightforward policy evaluation method. MC trains the function approximator towards the return, defined as $G(s) \doteq \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s$. We note that $\mathbb{E}[G(s)] = v(s)$ by definition. MC updates the weight vector as

$$w_{t+1} = w_t + \alpha_t (G(S_t) - \hat{v}_{w_t}(S_t)) \nabla \hat{v}_{w_t}(S_t),$$

where $\{\alpha_t\}$ is a sequence of learning rates. It is known that MC directly minimizes the MSVE [Sutton and Barto, 2018].

TD [Sutton, 1988, Sutton and Barto, 2018] is another fundamental policy evaluation algorithm. With linear function approximation, TD updates the weight vector iteratively as

$$\begin{aligned} w_{t+1} &= w_t + \alpha_t (R_{t+1} + \gamma \hat{v}_{w_t}(S_{t+1}) - \hat{v}_{w_t}(S_t)) \nabla \hat{v}_{w_t}(S_t) \\ &= w_t + \alpha_t \big(R_{t+1} + \gamma w_t^\top \phi(S_{t+1}) - w_t^\top \phi(S_t)\big) \phi(S_t), \end{aligned}$$

where $\{\alpha_t\}$ is a sequence of learning rates. The term $(R_{t+1} + \gamma \hat{v}_{w_t}(S_{t+1}) - \hat{v}_{w_t}(S_t))$ is the TD error responsible for correcting the estimate. Unlike MC, TD forms the training target using its estimate, known as "bootstrapping" [Sutton and Barto, 2018].

The norm of the expected update (NEU) loss is an objective that gradient TD methods [Sutton et al., 2008, 2009] optimize. While it was originally defined exclusively for linear function approximation, we generalize it in our work to arbitrary function approximation. Given the weights w, we define NEU as $\text{NEU}(w) \doteq \|\mathbb{E}[\delta \nabla \hat{v}_w(S)]\|$, where $\delta = G(S) - \hat{v}_w(S)$ for MC and $\delta = R + \gamma \hat{v}_w(S') - \hat{v}_w(S)$ for TD.

3.2 Linear Attention and Transformer

A canonical single-head self-attention [Vaswani et al., 2017] computes $\text{Attn}_{W_k, W_q, W_v}(Z) \doteq W_v Z \,\text{softmax}\left(Z^\top W_k^\top W_q Z\right)$, where W_k, W_q, W_v denote the key, query, and value matrices, respectively, and Z is the input prompt. The recent theoretical advancements of in-context reinforcement learning consider a linear attention variant, where the softmax activation is replaced with an identity mapping [Park et al., 2024, Wang et al., 2025]. We consider the same linear self-attention structure and define $\text{LinAttn}(Z; P, Q) \doteq PZM\left(Z^\top QZ\right)$, where $Z \in \mathbb{R}^{(2d+1) \times (n+1)}$ is the input prompt, $P \in \mathbb{R}^{(2d+1) \times (2d+1)}$ is the reparameterization of W_v, $Q \in \mathbb{R}^{(2d+1) \times (2d+1)}$ is the reparameterization of $W_k^\top W_q$, and $M \in \mathbb{R}^{(n+1) \times (n+1)}$ is a fixed mask defined as $M \doteq \begin{bmatrix} I_n & 0_{n \times 1} \\ 0_{1 \times n} & 0 \end{bmatrix}$. We delegate the first n columns of Z as the *context* and the $(n+1)$-th column for the *query*. The purpose of M is to separate the query from the context during forward propagation. Having defined the linear attention, we can define the linear Transformer as a stack of linear self-attention layers. In an L-layer Transformer with parameters $\{P_l, Q_l\}_{l=0,\ldots,L-1}$, the input prompt Z_0 evolves layer by layer as

$$Z_{l+1} \doteq Z_l + \text{LinAttn}_{P_l, Q_l}(Z_l) = Z_l + P_l Z_l M (Z_l^\top Q_l Z_l). \tag{1}$$

By the convention of Wang et al. [2025], we define $\mathrm{TF}_L(Z_0; \{P_l, Q_l\}_{l=0,1,\dots L-1}) \doteq -Z_L^{(2d+1,n+1)}$ as the output of the L-layer Transformer, given input prompt Z_0. Notably, $Z_L^{(2d+1,n+1)}$ is the bottom-right element of the output embedding matrix, which is a scalar. The negative sign is in place to be consistent with previous work and simplify the notations in the proof.

3.3 In-Context Temporal Difference Learning

Wang et al. [2025] show that under certain parameterization of P_l and Q_l, the layer-by-layer forward propagation of TF_L is equivalent to an iteration-by-iteration execution of a batched version of TD algorithm on the context. In particular, let $S_0, R_1, S_1, \dots, R_n, S_n$ be a trajectory unrolled from a task. Suppose we wish to estimate $v(s_q)$ for some query state $s_q \in \mathcal{S}$. Then, using shorthands $\phi_i \doteq \phi(S_i)$ and $\phi_q \doteq \phi(s_q)$, we define for $l = 0, 1, \dots, L-1$

$$Z_0 \doteq \begin{bmatrix} \phi_0 & \dots & \phi_{n-1} & \phi_q \\ \gamma\phi_1 & \dots & \gamma\phi_n & 0_{d\times 1} \\ R_1 & \dots & R_n & 0 \end{bmatrix} \in \mathbb{R}^{(2d+1)\times(n+1)}, \tag{2}$$

$$P_l^{\mathrm{TD}} \doteq \begin{bmatrix} 0_{2d\times 2d} & 0_{2d\times 1} \\ 0_{1\times 2d} & 1 \end{bmatrix}, Q_l^{\mathrm{TD}} \doteq \begin{bmatrix} -I_d & I_d & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0_{1\times d} & 0 \end{bmatrix} \in \mathbb{R}^{(2d+1)\times(2d+1)}.$$

Let $\theta_L^{\mathrm{TD}} \doteq \{P_l^{\mathrm{TD}}, Q_l^{\mathrm{TD}}\}_{l=0,\dots,L-1}$ represent the parameters compactly. Theorem 1 of Wang et al. [2025] proves that, given Z_0 defined in (2), it holds that $\mathrm{TF}_L(Z_0; \theta_L^{\mathrm{TD}}) = \langle \phi_q, w_l \rangle$, where $\{w_l\}$ is defined as $w_0 = 0$ and

$$w_{l+1} = w_l + \sum_{i=0}^{n-1} \left(R_{i+1} + \gamma w_l^\top \phi_{i+1} - w_l^\top \phi_i\right) \phi_i. \tag{3}$$

4 Inference Time Convergence

In this section, we establish the convergence and inference optimality of ICTD implemented by the L-layer linear Transformer as $L \to \infty$. Before we proceed to our analysis, we first need to redefine the input prompt Z_0 and the feature function ϕ to facilitate our proof in the rest of the paper. Let $n = |\mathcal{S}|$ and $s_0, s_1, \dots, s_{n-1}$ be the states in $\mathcal{S}$. We employ a one-hot feature function $\phi : \mathcal{S} \to \{0,1\}^n$ that maps each state s to an n-dimensional vector where the n-th dimension is one and zero everywhere else. Using shorthands $\phi_i \doteq \phi(s_i)$ and $R_{i+1} \doteq r(s_i)$, we redefine Z_0 as

$$Z_0 \doteq \begin{bmatrix} \phi_0 & \dots & \phi_{n-1} & \phi_q \\ \gamma \sum_{j=0}^{n-1} \phi_j p(s_j \mid s_0) & \dots & \gamma \sum_{j=0}^{n-1} \phi_j p(s_j \mid s_{n-1}) & 0_{n\times 1} \\ R_1 & \dots & R_n & 0 \end{bmatrix} \in R^{(2n+1)\times(n+1)}. \tag{4}$$

Now, the top n rows of Z_0 are the *enumerations* of the states in $\mathcal{S}$, represented in one-hot encodings. Rows n to $2n$ are now *expected* next features w.r.t p, discounted by γ. Hence, given Z_0 defined in (4), it holds that $\mathrm{TF}_L(Z_0; \theta_L^{\mathrm{TD}}) = \langle \phi_q, w_l \rangle$, where $\{w_l\}$ is defined as $w_0 = 0$ and

$$w_{l+1} = w_l + \sum_{i=0}^{n-1} \left(R_{i+1} + \gamma w_l^\top \sum_{j=0}^{n-1} \phi_j p(s_j \mid s_i) - w_l^\top \phi_i \right) \phi_i \tag{5}$$

by the virtue of (3). We employ $P \in [0,1]^{n\times n}$ to denote the transition probability matrix corresponding to p, where $P(i,j) \doteq p(s_j \mid s_i)$, and $\Phi \in \{0,1\}^{n\times n}$ to indicate the feature matrix, where the i-th row equals $\phi(s_i)$. It is obvious that $\Phi = I_n$. Likewise, we use $r \in \mathbb{R}^n$ to denote the vector representation of the reward function when it does not introduce ambiguity, where the i-th dimension is $r(s_i)$. Then, we have the following theorem.

Theorem 1. *Given a query state $s_q \in \mathcal{S}$ and constructing Z_0 as (4), it holds that $\lim_{L\to\infty} TF_L\left(Z_0; \theta_L^{TD}\right) = v(s_q)$.*

The proof is in Appendix A.1. Theorem 1 demonstrates that, under our formulation of Z_0 in (4), $\mathrm{TF}_L(Z_0, \theta_L^{\mathrm{TD}})$ converges to the *true value* $v(s_q)$ of the query state s_q as the Transformer gets infinitely deep. This finding hints at the superiority of the parameterization θ_L^{TD} from the policy evaluation point of view.

5 Global Minimizer of Pretraining Loss

Wang et al. [2025] discover that θ_L^{TD} emerges after pretraining on some task and feature distributions using the proposed multi-task TD update. Their effort in explaining the emergence of θ_L^{TD} includes proof that θ_L^{TD} belongs to an invariant set of the multi-task TD update. However, their analysis is restricted to the single-layer case, yet the phenomenon is also observed in multi-layer linear Transformers. Previously, we justify that θ_L^{TD} is desirable for in-context policy evaluation in Theorem 1. In this section, we investigate it from an optimization perspective and show that θ_L^{TD} is one of the global minimizers of the pretraining loss.

Let $\Delta(r)$ denote the distribution of reward functions and $\Delta(p)$ denote the distribution of transition probabilities. Then, the tuple $\langle \Delta(r), \Delta(p) \rangle$ characterizes the MRP distribution. Let θ denote the (vectorization and concatenation of) parameters $\{(P_l, Q_l)\}_{l=0,\dots,L-1}$ of a Transformer of L layers. We recall that in (4), the prompt Z_0 depends on the query feature ϕ_q, as well as p and r. To this end, we write Z_0 as $Z_0(p, r, \phi_q)$ to emphasize this dependency and use $Z_0(p, r, \phi(s))$ to denote the prompt by replacing the query ϕ_q with the feature $\phi(s)$ for some $s \in \mathcal{S}$. Wang et al. [2025] update θ iteratively as $\theta_{k+1} = \theta_k + \alpha_k \Delta^{\text{TD}}(\theta_k)$, where

$$\Delta^{\text{TD}}(\theta) \doteq \mathbb{E}_{s_q \sim \mu^p, s_q' \sim p(s_q), p \sim \Delta(p), r \sim \Delta(r)} [(r(s_q) + \gamma \text{TF}_L(Z_0'; \theta) - \text{TF}_L(Z_0; \theta)) \nabla_\theta \text{TF}_L(Z_0; \theta)].$$

Here, μ^p denotes the stationary distribution of the transition dynamics p. We further define $Z_0' \doteq Z_0(p, r, \phi(s_q'))$ and $Z_0 \doteq Z_0(p, r, \phi(s_q))$. This motivates us to consider the Norm of Expected Update (NEU) loss, defined as $J(\theta) \doteq \left\| \Delta^{\text{TD}}(\theta) \right\|_1$. Similar to Gatmiry et al. [2024], we enforce a sparsity assumption on the parameter space. Namely, we search θ over Θ, defined as

$$\Theta \doteq \left\{ \left(P = \begin{bmatrix} 0_{2n \times 2n} & 0_{2n \times 1} \\ u & 1 \end{bmatrix}, Q = \begin{bmatrix} A & 0_{2n \times 1} \\ 0_{1 \times 2n} & 0 \end{bmatrix} \right) \middle| u \in \mathbb{R}^{1 \times 2n}, A \in \mathbb{R}^{2n \times 2n} \right\}^L.$$

Notably, in the parameter space Θ, different Transformer layers can have different parameters (i.e., different u and A). One special case of Θ is called the looped (autoregressive) Transformer, where all Transformer layers are forced to share the same parameters. We use Θ^{Looped}, defined as

$$\Theta^{\text{Looped}} \doteq \left\{ \left(P = \begin{bmatrix} 0_{2n \times 2n} & 0_{2n \times 1} \\ u & 1 \end{bmatrix}, Q = \begin{bmatrix} A & 0_{2n \times 1} \\ 0_{1 \times 2n} & 0 \end{bmatrix} \right)^L \middle| u \in \mathbb{R}^{1 \times 2n}, A \in \mathbb{R}^{2n \times 2n} \right\},$$

to denote the parameter space (with sparsity constraints) of the looped Transformer we consider. It is easy to see that both $\theta_L^{\text{TD}} \in \Theta$ and $\theta_L^{\text{TD}} \in \Theta^{\text{Looped}}$ hold. Although the evaluation of $\text{TF}_L(Z_0; \theta)$ and $\text{TF}_L(Z_0'; \theta)$ does not depend on the choice of the parameter space, the evaluation of $\nabla_\theta \text{TF}_L(Z_0; \theta)$ does. When $\theta \in \Theta$, the gradient is taken w.r.t. L different pairs of (u, A). On the other hand, when $\theta \in \Theta^{\text{Looped}}$, the gradient is take w.r.t. a single pair of (u, A).

The theorem below confirms that θ_L^{TD} is the global optimizer of the NEU loss for the looped Transformer when the number of layers grows to ∞.

Theorem 2. *Let the parameter space be Θ^{Looped}. Then, it holds that* $\lim_{L \to \infty} J\left(\theta_L^{TD}\right) = 0$.

A few lemmas are in the sequl to prepare us to prove the above theorem.

Lemma 1. *For any query state s_q, transition probability p and reward function r, it holds that* $\left| \mathbb{E}_{s_q' \sim p(s_q)} \left[r(s_q) + \gamma TF_L\left(Z_0'; \theta_L^{TD}\right) - TF_L\left(Z_0; \theta_L^{TD}\right) \right] \right| \leq \|r\|_\infty \gamma^L$.

The proof is in Appendix A.2. Lemma 1 shows that the absolute value of the expected TD error decays at an exponential rate.

Define $X_l \doteq Z_l^{(1:2n,1:n)} \in \mathbb{R}^{2n \times n}$, the top left $2n$ by n block of Z_l, and $Y_l \doteq Z_l^{(2n+1,1:n)} \in \mathbb{R}^{1 \times n}$, the bottom row of Z_l except for the $(n+1)$-th entry to simplify notations. We furtherm define $x_l^q \doteq \begin{bmatrix} \phi_q \\ 0_{n \times 1} \end{bmatrix} \in \mathbb{R}^{2n}$ to represent the last column of Z_l except for the last dimension. Likewise, we write y_l^q to denote the bottom right entry of Z_l. Then, we can write Z_l as $Z_l = \begin{bmatrix} X_l & x_l^q \\ Y_l & y_l^q \end{bmatrix}$.

Lemma 2. *Let Z_l be the l-th embedding evolved by an L-layer linear Transformer parameterized by θ_L^{TD} following (1), where Z_0 is the input prompt defined in (4). Then, for $l = 0, 1, \ldots, L-1$, it holds that 1. $|y_l^q| \leq (1+\gamma^l)\|v\|_\infty$; 2. $\|Y_l\|_1 \leq (\gamma+\gamma^l)\|v\|_\infty$ for all query state $s_q \in \mathcal{S}$, transition probability p and reward function r.*

We leave the proof in Appendix A.3. Essentially, Lemma 2 implies that the norm of the evolving part of Z_l, i.e., the norm of Y_l and y_l^q, is uniformly bounded because $\gamma^l \leq 1$ for $l \geq 0$.

Lemma 3. $\left\|\nabla_\theta TF_L\left(Z_0; \theta_L^{TD}\right)\right\|_1 \leq \frac{L^2+L}{2}\nu + L\xi$ *for some constants ν and ξ independent of L.*

The proof can be found in Appendix A.4. Lemma 3 shows the norm of the gradient term grows at a polynomial rate. We now have everything we need to prove Theorem 2 in Appendix A.5.

Recall that in multi-task TD, Wang et al. [2025] use the target $r(s_q) + \gamma \mathrm{TF}_L(Z_0', \theta)$ to update the parameters of the Transformer, where $\mathrm{TF}_L(Z_0', \theta)$ can be viewed as a (biased) proxy for $v(s_q')$. As $\mathbb{E}\left[r(s_q) + \gamma v(s_q')\right] = v(s_q)$, we conjecture that using the Monte Carlo return $G(s_q)$ as the target can also enable in-context policy evaluation capabilities in Transformers. In light of this, we propose the multi-task MC update that updates θ iteratively as $\theta_{k+1} = \theta_k + \alpha_k \Delta^{\mathrm{MC}}(\theta_k)$, where

$$\Delta^{\mathrm{MC}}(\theta) \doteq \mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)}[(G(s_q) - \mathrm{TF}_L(Z_0; \theta))\nabla_\theta \mathrm{TF}_L(Z_0; \theta)].$$

We again consider the NEU loss, defined as $J'(\theta) \doteq \left\|\Delta^{\mathrm{MC}}(\theta)\right\|_1$. Remarkably, we also prove that θ_L^{TD} is the global optimizer of the NEU loss for our multi-task MC update as the Transformer's depth grows to infinity.

Corollary 1. *Let the parameter space be Θ^{Looped}. Then it holds that* $\lim_{L\to\infty} J'(\theta_L^{TD}) = 0$.

We need the following lemma to assist us in proving the corollary.

Lemma 4. *For any query state s_q, transition probability p and reward function r, it holds that* $\left|\mathbb{E}_{p,r}\left[G(s_q) - TF_L\left(Z_0; \theta_L^{TD}\right)\right]\right| \leq \|v\|_\infty \gamma^L$.

The proof is in Appendix A.6. Lemma 4 is analogous to Lemma 1 in that it proves the norm of the expected approximation error of MC decays at an exponential rate. We can now prove Corollary 1 in Appendix A.7.

6 Experiments

We conduct empirical studies to verify our theoretical results. In particular, mirroring our theoretical setup, we investigate the following two questions:

1. Does the pretraining yield a Transformer that can perform in-context policy evaluation?
2. Do the converged parameters align with θ^{TD}?

We answer both questions for both multi-task TD and multi-task MC.

Following Wang et al. [2025], we employ Boyan's chain [Boyan, 1999] as our environment to construct the MRPs. Boyan's chain allows us to have full information and control over the environment, including analytically solving for the true values and the stationary distributions. Figure 3 shows an example of an S-state Boyan's chain. Adapting the technique of Wang et al. [2025], we randomly generate the reward and transition probability functions, preserving the topology of the chain to ensure its ergodicity. Our only simplifying modification is setting the stationary distribution as the initial distribution. The details of the task generation, including the distributions to sample p and r, can be found in Algorithm 1.

Since our convergence results require $L \to \infty$, we construct a looped Transformer TF_L with $L = 30$. We find this depth yields consistent results without running into numerical instability. For each trial of the experiment, we generate 20,000 tasks for training with $\gamma = 0.9$. Within each task characterized by p, r, we generate a mini-batch of size $b = 64$. For multi-task TD, we unroll the MRP to generate a trajectory $S_0, R_1, S_1, R_2, \ldots, S_{b-1}, R_b, S_b$. Then, we update the parameter θ of the Transformer by

$$\theta_{k+1} = \theta_k + \frac{\alpha}{b}\sum_{i=0}^{b-1} \delta_i \nabla_\theta \mathrm{TF}_L(Z_0(p, r, \phi(S_i)); \theta_k),$$

where $\delta_i \doteq R_{i+1} + \gamma \text{TF}_L(Z_0(p, r, \phi(S_{i+1})); \theta_k) - \text{TF}_L(Z_0(p, r, \phi(S_i)); \theta_k)$. Similarly, for multi-task MC, we unroll the MRP to generate a trajectory $S_0, R_1, S_1, R_2, \ldots, S_{b-2}, R_{b-1}, S_{b-1}$. For each S_i, we independently unroll the MRP for 200 steps to obtain a truncated return $\tilde{G}(S_i)$ beginning from that state. Admittedly, $\tilde{G}(S_i)$ introduces bias due to truncation. However, the bias is negligible as $\gamma^{200} \approx 7 \times 10^{-10}$. Multi-task MC updates the parameter by

$$\theta_{k+1} = \theta_k + \frac{\alpha}{b} \sum_{i=0}^{b-1} \Big(\tilde{G}(S_i) - \text{TF}_L(Z_0(p, r, \phi(S_i)); \theta_k) \Big) \nabla_\theta \text{TF}_L(Z_0(p, r, \phi(S_i)); \theta_k).$$

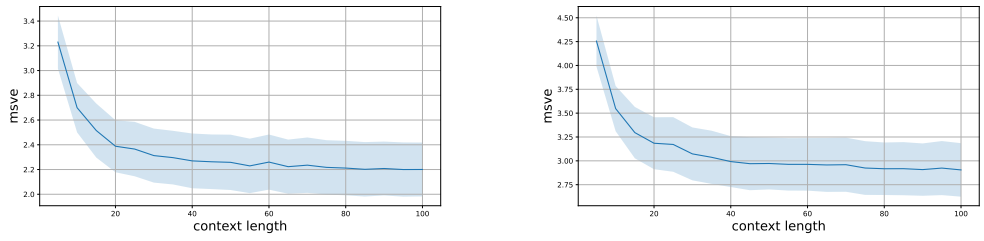
We inherit most of the experimental settings from Wang et al. [2025]. See Table 1 for a comprehensive list of hyperparameters and training details.

6.1 In-Context Policy Evaluation Verification

Recall that one property of ICRL is that the agent's performance improves as the context gets longer without parameter updates. Based on this principle, we investigate the MSVE as a dependent variable of the context length for the converged Transformers. After each trial, we sample $k = 10$ novel tasks $T_1, \ldots, T_k$ from the task distribution as our validation set. Notably, since the context length m here is dynamic, we can no longer enumerate the states from s_0 to s_{n-1} to form the context as in (4). Instead, we use sampled states to form the features in the top n rows. Regarding rows $n+1$ to $2n$, we still use the expected next feature. Thus, suppose we need a context with length m from T_i, we unroll T_i for m steps and get $S_0^{(i)}, R_1^{(i)}, S_1^{(i)}, \ldots, S_{m-2}^{(i)}, R_{m-1}^{(i)}, S_{m-1}^{(i)}, R_m^{(i)}$. Then, we form the context as

$$C(i, m) \doteq \begin{bmatrix} \phi\left(S_0^{(i)}\right) & \cdots & \phi\left(S_{m-1}^{(i)}\right) \\ \gamma \sum_{s' \in \mathcal{S}} p^{(i)}\left(s' \middle| S_0^{(i)}\right) \phi(s') & \cdots & \gamma \sum_{s' \in \mathcal{S}} p^{(i)}\left(s' \middle| S_{m-1}^{(i)}\right) \phi(s') \\ R_1^{(i)} & \cdots & R_m^{(i)} \end{bmatrix} \in \mathbb{R}^{(2n+1) \times m}.$$

Given a query state s_q, we then define the input prompt $Z(i, m, s_q) \doteq [C(i, m) \quad x_q] \in \mathbb{R}^{(2n+1) \times (m+1)}$, where $x_q = \begin{bmatrix} \phi(s_q)^\top & 0_{1 \times (n+1)} \end{bmatrix}^\top$. Then, given a parameter θ, we can evaluate the MSVE of the Transformer on T_i with context length m by $\text{MSVE}(i, m) \doteq \sum_{s \in \mathcal{S}} \mu^{p^{(i)}}(s) (\text{TF}_L(Z(i, m, s); \theta) - v(s))^2$, where $\mu^{p^{(i)}}$ means the stationary distribution with respect to the transition probabilities of T_i. We average the MSVEs across the validation tasks to get an accurate estimation of the expected MSVE with context length m. We compute the averaged MSVEs with respect to an array of increasing m's. We repeat our experiments for 20 trials for both multi-task TD and MC using the last checkpoint after pretraining. Figure 1a and 1b plot the means and standard errors of the averaged MSVEs against the context lengths for multi-task TD and MC, respectively. The MSVEs exhibit a clear decreasing trend as the context gets longer for both pretraining schemes. Therefore, we can confidently conclude that multi-task TD and MC can produce Transformers capable of in-context policy evaluation. The answer to question 1 is affirmative.



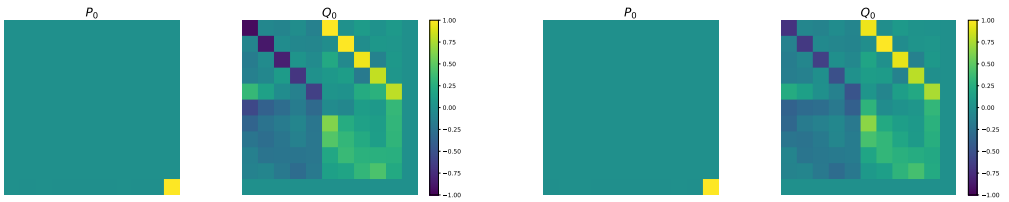
(a) Multi-task TD (b) Multi-task MC

Figure 1: Mean and standard error of the averaged MSVEs against context lengths. The curves are averaged over 20 random trials. The shaded areas represent the standard errors.

6.2 Weight Convergence

In this section, we inspect the parameters of the Transformers after pretraining with multi-task TD and MC. Figure 2 displays the final P and Q matrices after pretraining. Despite mild noise, both

methods produce the parameter configuration θ_L^{TD}. Hence, the answer to question 2 is also affirmative. Surprisingly, even their noise patterns look very similar. Therefore, we speculate that the weights converge to the same place with multi-task TD and MC. Notably, the Transformer learns to implement in-context TD even after pretraining with multi-task MC. This observation testifies to the fact that the model does not merely imitate the reinforcement pretraining algorithm like in algorithm distillation for supervised pretraining. Instead, the model itself decides how to best learn from the context.



(a) Multi-task TD (b) Multi-task MC

Figure 2: Mean Transformer parameters after pretraining. The parameters are averaged over 20 trials and normalized to stay in the range $[-1, 1]$.

7 Limitations and Future Works

While our work is the first to dive deeper into why reinforcement pretraining enables ICTD, it has several limitations. Firstly, our analysis assumes a linear Transformer. Although linear attentions are usual in theoretical analysis for in-context learning [Ahn et al., 2024, Gatmiry et al., 2024, Park et al., 2024, Wang et al., 2025], it leaves a gap between theory and practice. Future works can look into extending the results to simple non-linear activations, such as ReLU. Secondly, our prompt construction assumes an enumeration of the states and expected next features. While this assumption facilitates our analysis, it can hardly hold in practice, especially in MDPs with a continuous state space. Future works may include generalizing our results to sampled states and/or next features. Thirdly, our empirical study is relatively small in scale. While it is intended for a clear proof of concept, future work can scale up our experiments. In this work, we focus only on the prediction problem. Exciting future directions include studying the emergence of in-context control via reinforcement pretraining.

8 Conclusion

This work is the first in-depth analysis of the emergence of the in-context TD parameter through reinforcement pretraining. We show that under some prompt construction, the in-context prediction of the Transformer with ICTD weights converges to the true value as it gets deeper. Furthermore, we prove that the ICTD parameter is one minimizer of the NEU loss for both multi-task TD and MC pretraining as the number of attention layers goes to infinity. Through controlled experiments, we show that (1) both multi-task TD and MC induce in-context policy evaluation capabilities in linear Transformers, and (2) both pretraining schemes produce the same final parameter — the ICTD weights. We acknowledge the limitations of our work, including the linear attention assumption put on the Transformer, the practicality issue of our prompt construction, and the small scale of our empirical study. We believe addressing these limitations will make admirable contributions to the advancement of ICRL.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In Advances in Neural Information Processing Systems, 2024.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao,

Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2024.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2017.

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, et al. Human-timescale adaptation in an open-ended task space. In Proceedings of the International Conference on Machine Learning, 2023.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. ArXiv Preprint, 2023.

Justin A. Boyan. Least-squares temporal difference learning. In Proceedings of the International Conference on Machine Learning, 1999.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Advances in Neural Information Processing Systems, 2020.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In International Conference on Learning Representations, 2021.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011.

Zhenwen Dai, Federico Tomasi, and Sina Ghiassian. In-context exploration-exploitation for reinforcement learning. In The Twelfth International Conference on Learning Representations, 2024.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: fast reinforcement learning via slow reinforcement learning. ArXiv Preprint, 2016.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning?, 2024.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. ArXiv Preprint, 2023.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Ikechukwu D Adebi, Amy Zhang, and Yuke Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. In Advances in Neural Information Processing Systems, 2024.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. Nature, 2020.

Sili Huang, Jifeng Hu, Hechang Chen, Lichao Sun, and Bo Yang. In-context decision transformer: Reinforcement learning via hierarchical chain-of-thought. ArXiv Preprint, 2024a.

Sili Huang, Jifeng Hu, Zhejian Yang, Liwei Yang, Tao Luo, Hechang Chen, Lichao Sun, and Bo Yang. Decision mamba: Reinforcement learning via hybrid selective sequence modeling. ArXiv Preprint, 2024b.

J. D. Hunter. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 2007.

JGraph. draw.io, October 2021. URL https://github.com/jgraph/drawio.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, 2015.

Louis Kirsch, Sebastian Flennerhag, Hado van Hasselt, Abram Friesen, Junhyuk Oh, and Yutian Chen. Introducing symmetries to black box meta reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 2022.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, maxime gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In Proceedings of the International Conference on Learning Representations, 2023.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. ArXiv Preprint, 2023.

Hao Liu and Pieter Abbeel. Emergent agentic transformer from chain of hindsight experience. In Proceedings of the 40th International Conference on Machine Learning, 23–29 Jul 2023.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. In Advances in Neural Information Processing Systems, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2016.

Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shangtong Zhang. A survey of in-context reinforcement learning. ArXiv Preprint, 2025.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. Do llm agents have regret? a case study in online learning and games. ArXiv Preprint, 2024.

Martin L Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. Foundations and Trends® in Machine Learning, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. ArXiv Preprint, 2017.

Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. Machine Learning, 2007.

Lucy Xiaoyang Shi, Yunfan Jiang, Jake Grigsby, Linxi Fan, and Yuke Zhu. Cross-episodic curriculum for transformer agents. In Advances in Neural Information Processing Systems, 2024.

Richard S. Sutton. Learning to predict by the methods of temporal differences. Machine Learning, 1988.

Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction (2nd Edition). MIT press, 2018.

Richard S. Sutton, Csaba Szepesvári, and Hamid Reza Maei. A convergent O(n) temporal-difference algorithm for off-policy learning with linear function approximation. In Advances in Neural Information Processing Systems, 2008.

Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In Proceedings of the International Conference on Machine Learning, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, 2017.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. ArXiv Preprint, 2016.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning. In Proceedings of the International Conference on Learning Representations, 2025.

Tengye Xu, Zihao Li, and Qinyuan Ren. Meta-reinforcement learning robust to distributional shift via performing lifelong in-context learning. In Proceedings of the International Conference on Machine Learning, 2024.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. Journal of Machine Learning Research, 2024.

Ilya Zisman, Vladislav Kurenkov, Alexander Nikulin, Viacheslav Sinii, and Sergey Kolesnikov. Emergence of in-context reinforcement learning from noise distillation. ArXiv Preprint, 2023.

A Proof of Theoretical Results

A.1 Theorem 1

Proof. We can rewrite (5) compactly as

$$\begin{aligned} w_{l+1} =& w_l + \Phi^\top r + \gamma \Phi^\top P \Phi w_l - \Phi^\top \Phi w_l \\ =& w_l + \Phi^\top (r + (\gamma P - I)\Phi w_l). \end{aligned}$$

Substituting $\Phi = I_n$, we have

$$\begin{aligned} w_{l+1} =& w_l + r + (\gamma P - I) w_l \\ =& r + \gamma P w_l. \end{aligned}$$

Unrolling the recursive definition, we get

$$\begin{aligned} w_0 =& 0 \\ w_1 =& r \\ w_2 =& r + \gamma P r \\ w_3 =& r + \gamma P r + (\gamma P)^2 r \\ & \vdots \\ w_l =& \sum_{k=0}^{l-1} (\gamma P)^k r. \end{aligned} \tag{6}$$

Therefore, we have

$$\lim_{L\to\infty} w_L = \lim_{L\to\infty} \sum_{k=0}^{L-1} (\gamma P)^k r = (I - \gamma P)^{-1} r.$$

With a slight abuse of notation, we employ $v \in \mathbb{R}^n$ to denote the value function in vector form, where the s-th dimension is $v(s)$. When writing out v as a Bellman equation, we have

$$\begin{aligned} v &= r + \gamma P v \\ (I - \gamma P) v &= r \\ v &= (I - \gamma P)^{-1} r. \end{aligned} \tag{7}$$

Thus, we have shown that $\lim_{L\to\infty} w_L = v$.

Finally, recall that

$$\mathrm{TF}_L(Z_0; \theta_L^{\mathrm{TD}}) = \langle \phi_q, w_L \rangle.$$

We therefore have

$$\lim_{L\to\infty} \mathrm{TF}_L(Z_0; \theta_L^{\mathrm{TD}}) = \lim_{L\to\infty} \langle \phi_q, w_L \rangle = \left\langle \phi_q, \lim_{L\to\infty} w_L \right\rangle = \langle \phi_q, v \rangle = v(s_q).$$

□

A.2 Lemma 1

Proof.

$$
\begin{aligned}
&\left|\mathbb{E}_{s'_q\sim p(s_q)}\left[r(s_q)+\gamma \mathrm{TF}_L\left(Z'_0;\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right]\right| \\
=&\left|r(s_q)+\gamma\mathbb{E}_{s'_q\sim p(s_q)}\left[\mathrm{TF}_L\left(Z'_0;\theta_L^{\mathrm{TD}}\right)\right]-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right| \\
=&\left|\langle\phi_q,r\rangle+\gamma\mathbb{E}_{s'_q\sim p(s_q)}\left[\left\langle\phi'_q,w_L\right\rangle\right]-\langle\phi_q,w_L\rangle\right| \\
=&\left|\phi_q^\top r+\gamma\mathbb{E}_{s'_q\sim p(s_q)}[\phi'_q]^\top\sum_{k=0}^{L-1}(\gamma P)^k r-\phi_q^\top\sum_{k=0}^{L-1}(\gamma P)^k r\right| && \text{(eq. (6))} \\
=&\left|\phi_q^\top r+\gamma\left(\phi_q^\top P\right)\sum_{k=0}^{L-1}(\gamma P)^k r-\phi_q^\top\sum_{k=0}^{L-1}(\gamma P)^k r\right| \\
=&\left|\phi_q^\top r+\phi_q^\top\sum_{k=1}^{L}(\gamma P)^k r-\phi_q^\top\sum_{k=0}^{L-1}(\gamma P)^k r\right| \\
=&\left|\phi_q^\top r+\phi_q^\top(\gamma P)^L r-\phi_q^\top r\right| \\
=&\gamma^L\left|\left\langle\phi_q,P^L r\right\rangle\right| \\
\le&\gamma^L\|\phi_q\|_1\left\|P^L r\right\|_\infty && \text{(Hölder's inequality)} \\
=&\gamma^L\left\|P^L r\right\|_\infty && (\|\phi_q\|_1=1) \\
\le&\gamma^L\left\|P^L\right\|_\infty\|r\|_\infty \\
=&\gamma^L\|r\|_\infty && (\left\|P^L\right\|_\infty=1)
\end{aligned}
$$

□

A.3 Lemma 2

Proof. We first note that $y_l^q=-\mathrm{TF}_L(Z_0;\theta_L^{\mathrm{TD}})=-\langle\phi_q,w_l\rangle$. Therefore, we get

$$
\begin{aligned}
|y_l^q|=&|\langle\phi_q,w_l\rangle| \\
\le&\|\phi_q\|_1\|w_l\|_\infty && \text{(Hölder's inequality)} \\
=&\|w_l\|_\infty && (\|\phi_q\|_1=1) \\
=&\left\|\sum_{k=0}^{l-1}(\gamma P)^k r\right\|_\infty && \text{(eq. (6))} \\
=&\left\|\left(I-(\gamma P)^l\right)(I-\gamma P)^{-1}r\right\|_\infty \\
=&\left\|\left(I-(\gamma P)^l\right)v\right\|_\infty && \text{(eq. (7))} \\
=&\left\|v-\gamma^l P^l v\right\|_\infty \\
\le&\|v\|_\infty+\left\|\gamma^l P^l v\right\|_\infty \\
\le&\|v\|_\infty+\gamma^l\left\|P^l\right\|_\infty\|v\|_\infty \\
=&(1+\gamma^l)\|v\|_\infty && (\left\|P^l\right\|_\infty=1)
\end{aligned}
$$

We have proved 1. of Lemma 2.

Under the structure of P_l in θ_L^{TD}, it holds that the first $2n$ rows of Z_l remain unchanged. Thus, we drop the subscript of X_l and write

$$
Z_l=\begin{bmatrix}X & x^q\\ Y_l & y_l^q\end{bmatrix}.
$$

We then proceed to compute the recursive forms of Y_l and y_l^q. We first have

$$
P_l Z_l M=\begin{bmatrix}0_{2n\times 2n} & 0_{2n\times 1}\\ 0_{1\times 2n} & 1\end{bmatrix}\begin{bmatrix}X & x^q\\ Y_l & y_l^q\end{bmatrix}\begin{bmatrix}I_n & 0_{n\times 1}\\ 0_{1\times n} & 0\end{bmatrix}
$$

$$= \begin{bmatrix} 0_{2n\times 2n} & 0_{2n\times 1} \\ 0_{1\times 2n} & 1 \end{bmatrix} \begin{bmatrix} X & 0_{2n\times 1} \\ Y_l & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0_{2n\times n} & 0_{2n\times 1} \\ Y_l & 0 \end{bmatrix}.$$

Next, we have

$$Z_l^\top Q_l Z_l = \begin{bmatrix} X^\top & Y_l^\top \\ {x^q}^\top & y_l^q \end{bmatrix} \begin{bmatrix} A_l & 0_{2n\times 1} \\ 0_{1\times 2n} & 0 \end{bmatrix} \begin{bmatrix} X & x^q \\ Y_l & y_l^q \end{bmatrix}$$
$$= \begin{bmatrix} X^\top & Y_l^\top \\ {x^q}^\top & y_l^q \end{bmatrix} \begin{bmatrix} A_l X & A_l x^q \\ 0_{1\times n} & 0 \end{bmatrix}$$
$$= \begin{bmatrix} X^\top A_l X & X^\top A_l x^q \\ {x^q}^\top A_l X & {x^q}^\top A_l x^q \end{bmatrix}.$$

Putting the two parts together, we have

$$P_l Z_l M\left(Z_l^\top Q_l Z_l\right) = \begin{bmatrix} 0_{2n\times n} & 0_{2n\times 1} \\ Y_l & 0 \end{bmatrix} \begin{bmatrix} X^\top A_l X & X^\top A_l x^q \\ {x^q}^\top A_l X & {x^q}^\top A_l x^q \end{bmatrix}$$
$$= \begin{bmatrix} 0_{2n\times n} & 0_{2n\times 1} \\ Y_l X^\top A_l X & Y_l X^\top A_l x^q \end{bmatrix}.$$

We therefore get

$$\begin{cases} Y_{l+1} & = Y_l + Y_l X^\top A_l X; \\ y_{l+1}^q & = y_l^q + Y_l X^\top A_l x^q. \end{cases}$$

By induction, we can prove that

$$Y_{l+1} = Y_0 + \sum_{i=0}^{l} Y_i X^\top A_i X.$$

Similarly, we can show that

$$y_{l+1}^q = y_0^q + \sum_{i=0}^{l} Y_i X^\top A_i x^q.$$

Since $y_0^q = 0$, we get

$$y_{l+1}^q = \sum_{i=0}^{l} Y_i X^\top A_i x^q. \tag{8}$$

Let $y_l^{(k)}$ denote the k-th component of Y_l. It is clear that

$$y_{l+1}^{(k)} = y_0^{(k)} + \sum_{i=0}^{l} Y_i X^\top A_i x^{(k)}, \tag{9}$$

where $x^{(k)}$ denotes the k-th column of X. Recall that $y_l^q = -\langle \phi_q, w_l \rangle$. Substituting $x^{(k)}$ in (9) into x^q in (8), we get

$$y_l^{(k)} = y_0^{(k)} - \langle \phi_k, w_l \rangle.$$

Therefore, we have

$$Y_l = Y_0 - w_l^\top I_n = r^\top - w_l^\top.$$

We finally get

$$\begin{aligned} \|Y_l\|_1 =& \left\|Y_l^\top\right\|_\infty \\ =& \|r - w_l\|_\infty \end{aligned}$$

$$
\begin{aligned}
&= \left\| r - \sum_{k=0}^{l-1} (\gamma P)^k r \right\|_\infty \\
&= \left\| r - \left(I - (\gamma P)^l\right)(I - \gamma P)^{-1} r \right\|_\infty \\
&= \left\| r - \left(I - (\gamma P)^l\right) v \right\|_\infty && \text{(eq. (7))} \\
&= \left\| (I - \gamma P)(I - \gamma P)^{-1} r - \left(I - (\gamma P)^l\right) v \right\|_\infty \\
&= \left\| (I - \gamma P) v - \left(I - (\gamma P)^l\right) v \right\|_\infty \\
&= \left\| (\gamma P)^l v - \gamma P v \right\|_\infty \\
&\le \left\| (\gamma P)^l v \right\|_\infty + \left\| \gamma P v \right\|_\infty \\
&\le \gamma^l \left\| P^l \right\|_\infty \|v\|_\infty + \gamma \|P\|_\infty \|v\|_\infty \\
&= (\gamma^l + \gamma) \|v\|_\infty . && \left(\left\| P^l \right\|_\infty = 1, \|P\|_\infty = 1\right)
\end{aligned}
$$

This concludes the proof of 2. of Lemma 2. □

A.4 Lemma 3

Proof. Since $\theta_L^{\mathrm{TD}} \in \Theta^{\mathrm{Looped}}$, we can drop the subscripts of the linear Transformer parameters because of parameter sharing across the layers. We further have $\nabla_\theta \mathrm{TF}_L\left(Z_0; \theta_L^{\mathrm{TD}}\right) \doteq \begin{bmatrix} \mathrm{vec}\left(\nabla_A \mathrm{TF}_L\left(Z_0; \theta_L^{\mathrm{TD}}\right)\right) \\ \nabla_u^\top \mathrm{TF}_L\left(Z_0; \theta_L^{\mathrm{TD}}\right) \end{bmatrix} \in \mathbb{R}^{(2n+1)(2n)}$, where vec denotes the vectorization operation. We then have

$$
\begin{cases} Y_{l+1} &= Y_l + (uX + Y_l) X^\top A X = u X X^\top A X + Y_l \left(I + X^\top A X\right); \\ y_{l+1}^q &= y_l^q + (uX + Y_l) X^\top A x^q = y_l^q + u X X^\top A x^q + Y_l X^\top A x^q. \end{cases} \tag{10}
$$

We begin by solving for $\nabla_A y_{l+1}^{(i)}$, where $y_{l+1}^{(i)}$ denotes the i-th component of Y_{l+1}. Define $f(A) \doteq I + X^\top A X$ to simplify notations. By (10), we have

$$
\begin{aligned}
\nabla_A y_{l+1}^{(i)} &= \frac{\mathrm{d}\left[u X X^\top A X\right]^{(i)}}{\mathrm{d}A} + \frac{\mathrm{d}[Y_l f(A)]^{(i)}}{\mathrm{d}A} \\
&= \frac{\mathrm{d}\left[u X X^\top A x^{(i)}\right]}{\mathrm{d}A} + \frac{\mathrm{d}\left[\sum_{j=1}^n y_l^{(j)} f(A)_{j,i}\right]}{\mathrm{d}A},
\end{aligned}
$$

where $x^{(i)}$ indicates the i-th column of X. We analyze the summands separately. Since $\nabla_M \left(a^\top M b\right) = a b^\top$ given matrix M and vectors a and b, it holds that

$$
\frac{\mathrm{d}\left[u X X^\top A x^{(i)}\right]}{\mathrm{d}A} = X X^\top u^\top x^{(i)^\top}.
$$

Plugging in the condition that $u = 0$ in θ_L^{TD}, we get $\frac{\mathrm{d}\left[u X X^\top A x^{(i)}\right]}{\mathrm{d}A} = 0$.

Next, we tackle the gradient term $\frac{\mathrm{d}\left[\sum_{j=1}^n y_l^{(j)} f(A)_{j,i}\right]}{\mathrm{d}A}$. By the summation rule and the product rule, we get

$$
\frac{\mathrm{d}\left[\sum_{j=1}^n y_l^{(j)} f(A)_{j,i}\right]}{\mathrm{d}A} = \sum_{j=1}^n \frac{\mathrm{d}\left[y_l^{(j)} f(A)_{j,i}\right]}{\mathrm{d}A} = \sum_{j=1}^n \left(f(A)_{j,i} \frac{\mathrm{d}y_l^{(j)}}{\mathrm{d}A} + y_l^{(j)} \frac{\mathrm{d}f(A)_{j,i}}{\mathrm{d}A} \right). \tag{11}
$$

Employing the structure of A, we first note that

$$
\begin{aligned}
f(A) &= I + X^\top A X \\
&= I + \begin{bmatrix} I & \gamma P \end{bmatrix} \begin{bmatrix} -I & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I \\ \gamma P^\top \end{bmatrix} \\
&= I + \begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} I \\ \gamma P^\top \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= I + \gamma P^\top - I \\
&= \gamma P^\top.
\end{aligned}
$$

We also note that

$$
\frac{\mathrm{d} f(A)_{j,i}}{\mathrm{d}A} = \frac{\mathrm{d}\left[I_{j,i} + x^{(j)^\top} A x^{(i)}\right]}{\mathrm{d}A} = x^{(j)} x^{(i)^\top}.
$$

Therefore, plugging them in (11), we have

$$
\begin{aligned}
\frac{\mathrm{d}\left[\sum_{j=1}^n y_l^{(j)} f(A)_{j,i}\right]}{\mathrm{d}A} &= \sum_{j=1}^n \left(\gamma P_{i,j} \nabla_A y_l^{(j)} + y_l^{(j)} x^{(j)} x^{(i)^\top}\right) \\
&= X Y_l^\top x^{(i)^\top} + \sum_{j=1}^n \gamma P_{i,j} \nabla_A y_l^{(j)}.
\end{aligned}
$$

Consequently, we have

$$
\nabla_A y_{l+1}^{(i)} = X Y_l^\top x^{(i)^\top} + \sum_{j=1}^n \gamma P_{i,j} \nabla_A y_l^{(j)}.
$$

We now bound the gradient term norm $\left\|\nabla_A y_{l+1}^{(i)}\right\|_1$. We first have

$$
\begin{aligned}
\left\|\nabla_A y_{l+1}^{(i)}\right\|_1 &= \left\| X Y_l^\top x^{(i)^\top} + \sum_{j=1}^n \gamma P_{i,j} \nabla_A y_l^{(j)} \right\|_1 \\
&\le \left\| X Y_l^\top x^{(i)^\top} \right\|_1 + \left\| \sum_{j=1}^n \gamma P_{i,j} \nabla_A y_l^{(j)} \right\|_1 \\
&\le \|X\|_1 \left\|Y_l^\top\right\|_1 \left\|x^{(i)^\top}\right\|_1 + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1 \\
&= \left\| \begin{bmatrix} I \\ \gamma P^\top \end{bmatrix} \right\|_1 \|Y_l\|_\infty \left\|x^{(i)}\right\|_\infty + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1 \\
&= (1+\gamma)\|Y_l\|_\infty + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1 \\
&\le n(1+\gamma)(\gamma^l + \gamma)\|v\|_\infty + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1 \qquad \text{(Lemma 2)} \\
&\le n(1+\gamma)^2 \|v\|_\infty + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1.
\end{aligned}
$$

Let $\beta \doteq n(1+\gamma)^2\|v\|_\infty$. Having observed this pattern, we claim

$$
\left\|\nabla_A y_l^{(i)}\right\|_1 \le l\beta \tag{12}
$$

for all $i = 1, 2, \ldots, n$ and prove it by induction. Since Y_0 is not dependent on A, we have $\nabla_A Y_0 = 0$. Therefore, the base case $\left\|\nabla_A y_0^{(i)}\right\|_1 \le 0$ trivially holds. Then, suppose (12) holds for l. We have

$$
\begin{aligned}
\left\|\nabla_A y_{l+1}^{(i)}\right\|_1 &\le \beta + \gamma \sum_{j=1}^n P_{i,j} \left\|\nabla_A y_l^{(j)}\right\|_1 \\
&\le \beta + \gamma \sum_{j=1}^n P_{i,j} l\beta
\end{aligned}
$$

$$
\begin{aligned}
&= \beta + \gamma l \beta \\
&= (1 + \gamma l)\beta \\
&< (1 + l)\beta.
\end{aligned}
$$

The bound in (12) is proved.

We then proceed with $\nabla_A y_{l+1}^q$. Define $g(A) \doteq X^\top A x^q$ to simplify notations. Similar to $\nabla_A Y_l$, by (10), we have

$$
\begin{aligned}
\nabla_A y_{l+1}^q =& \nabla_A y_l^q + \frac{\mathrm{d}\left[u X X^\top A x^q\right]}{\mathrm{d}A} + \frac{\mathrm{d}[Y_l g(A)]}{\mathrm{d}A} \\
=& \nabla_A y_l^q + \frac{\mathrm{d}\left[u X X^\top A x^q\right]}{\mathrm{d}A} + \frac{\mathrm{d}\left[\sum_{k=1}^n y_l^{(k)} g(A)_k\right]}{\mathrm{d}A}.
\end{aligned}
$$

Firstly, we get

$$
\frac{\mathrm{d}\left[u X X^\top A x^q\right]}{\mathrm{d}A} = X X^\top u^\top {x^q}^\top.
$$

Substituting $u = 0$ into the equation, we get $\frac{\mathrm{d}\left[u X X^\top A x^q\right]}{\mathrm{d}A} = 0$. Then, we have

$$
\frac{\mathrm{d}\left[\sum_{k=1}^n y_l^{(k)} g(A)_k\right]}{\mathrm{d}A} = \sum_{k=1}^n \frac{\mathrm{d}\left[y_l^{(k)} g(A)_k\right]}{\mathrm{d}A} = \sum_{k=1}^n \left(g(A)_k \frac{\mathrm{d}y_l^{(k)}}{\mathrm{d}A} + y_l^{(k)} \frac{\mathrm{d}g(A)_k}{\mathrm{d}A} \right) \tag{13}
$$

by the summation rule and the product rule. We note that, by the structure of A in θ_L^{TD}, we have

$$
\begin{aligned}
g(A) =& X^\top A x^q \\
=& [I \quad \gamma P] \begin{bmatrix} -I & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi_q \\ 0 \end{bmatrix} \\
=& [I \quad \gamma P] \begin{bmatrix} -\phi_q \\ 0 \end{bmatrix} \\
=& -\phi_q.
\end{aligned} \tag{14}
$$

We further note that

$$
\frac{\mathrm{d}g(A)_k}{\mathrm{d}A} = \frac{\mathrm{d}\left[{x^{(k)}}^\top A x^q\right]}{\mathrm{d}A} = x^{(k)} {x^q}^\top.
$$

Combining these and plugging into (13), we get

$$
\begin{aligned}
\frac{\mathrm{d}\left[\sum_{k=1}^n y_l^{(k)} g(A)_k\right]}{\mathrm{d}A} =& \sum_{k=1}^n \left(y_l^{(k)} x^{(k)} {x^q}^\top - [\phi_q]_k \nabla_A y_l^{(k)} \right) \\
=& X {Y_l}^\top {x^q}^\top - \sum_{k=1}^n [\phi_q]_k \nabla_A y_l^{(k)}.
\end{aligned}
$$

Since ϕ_q is a one-hot vector, we let ω be the index where $[\phi_q]_\omega = 1$. We then have

$$
\frac{\mathrm{d}\left[\sum_{k=1}^n y_l^{(k)} g(A)_k\right]}{\mathrm{d}A} = X {Y_l}^\top {x^q}^\top - \nabla_A y_l^{(\omega)}.
$$

Hence, we have

$$
\nabla_A y_{l+1}^q = \nabla_A y_l^q + X {Y_l}^\top {x^q}^\top - \nabla_A y_l^{(\omega)}.
$$

Now, we derive the bound for $\left\| \nabla_A y_{l+1}^q \right\|_1$. We first have

$$
\left\| \nabla_A y_{l+1}^q \right\|_1 = \left\| \nabla_A y_l^q + X {Y_l}^\top {x^q}^\top - \nabla_A y_l^{(\omega)} \right\|_1
$$

$$\leq \|\nabla_A y_l^q\|_1 + \left\|XY_l^\top x^{q^\top}\right\|_1 + \left\|\nabla_A y_l^{(\omega)}\right\|_1.$$

We note that

$$\begin{aligned}\left\|XY_l^\top x^{q^\top}\right\|_1 &\leq \left\|\begin{bmatrix} I \\ \gamma P^\top \end{bmatrix}\right\|_1 \|Y_l^\top\|_1 \left\|x^{q^\top}\right\|_1 \\ &= (1+\gamma)\|Y_l\|_\infty \\ &\leq n(1+\gamma)(\gamma^l+\gamma)\|v\|_\infty \qquad \text{(Lemma 2)} \\ &\leq n(1+\gamma)^2\|v\|_\infty = \beta.\end{aligned}$$

Therefore, by (12), we have

$$\left\|\nabla_A y_{l+1}^q\right\|_1 \leq \|\nabla_A y_l^q\|_1 + (l+1)\beta.$$

With this form, we claim and prove by induction that

$$\|\nabla_A y_l^q\|_1 \leq \frac{l^2+l}{2}\beta \tag{15}$$

for $l \geq 0$. Since y_0^q does not depend on A, we have $\nabla_A y_0^q = 0$. Therefore, the base case $\|\nabla_A y_0^q\|_1 \leq 0$ trivially holds. Now suppose (15) holds for l. We get

$$\begin{aligned}\left\|\nabla_A y_{l+1}^q\right\|_1 &\leq \|\nabla_A y_l^q\|_1 + (l+1)\beta \\ &\leq \frac{l^2+l}{2}\beta + (l+1)\beta \\ &= \frac{(l+1)^2+(l+1)}{2}\beta.\end{aligned}$$

The bound in (15) is proved.

Next, we solve for $\nabla_u Y_{l+1}$. By (10), we first have

$$\begin{aligned}\nabla_u y_{l+1}^{(i)} =& \frac{\mathrm{d}\left[uXX^\top Ax^{(i)}\right]}{\mathrm{d}u} + \frac{\mathrm{d}\left[\gamma\sum_{j=1}^n P_{i,j} y_l^{(j)}\right]}{\mathrm{d}u} \\ =& XX^\top Ax^{(i)} + \gamma\sum_{j=1}^n P_{i,j}\nabla_u y_l^{(j)}.\end{aligned}$$

Define $\nabla_u Y_l \doteq \begin{bmatrix}\nabla_u y_l^{(1)} & \nabla_u y_l^{(2)} & \cdots & \nabla_u y_l^{(n)}\end{bmatrix} \in \mathbb{R}^{2n\times n}$. We thus have

$$\nabla_u Y_{l+1} = XX^\top AX + \gamma\nabla_u Y_l P^\top.$$

We now bound $\|\nabla_u Y_{l+1}\|_1$. We get

$$\begin{aligned}\|\nabla_u Y_{l+1}\|_1 &= \left\|XX^\top AX + \gamma\nabla_u Y_l P^\top\right\|_1 \\ &\leq \left\|XX^\top AX\right\|_1 + \left\|\gamma\nabla_u Y_l P^\top\right\|_1 \\ &\leq \|X\|_1\left\|X^\top AX\right\|_1 + \gamma\|\nabla_u Y_l\|_1\left\|P^\top\right\|_1 \\ &= \left\|\begin{bmatrix} I \\ \gamma P^\top \end{bmatrix}\right\|_1 \left\|[I \quad \gamma P]\begin{bmatrix} -I & I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} I \\ \gamma P^\top \end{bmatrix}\right\|_1 + \gamma\|\nabla_u Y_l\|_1 \\ &= (1+\gamma)\left\|\gamma P^\top - I\right\|_1 + \gamma\|\nabla_u Y_l\|_1 \\ &\leq (1+\gamma)^2 + \gamma\|\nabla_u Y_l\|_1.\end{aligned}$$

Let $\zeta \doteq (1+\gamma)^2$. We claim

$$\|\nabla_u Y_l\|_1 \leq l\zeta \tag{16}$$

and prove it by induction. We first have $\nabla_u Y_0 = 0$ because Y_0 does not depend on u. Hence, the base case $\|\nabla_u Y_0\|_1 \leq 0$ holds trivially. Assume (16) holds for l. We have

$$\|\nabla_u Y_{l+1}\|_1 \leq \zeta + \gamma\|\nabla_u Y_l\|_1$$

$$
\begin{aligned}
&\leq \zeta + \gamma l\zeta \\
&\leq (1+\gamma l)\zeta \\
&< (1+l)\zeta.
\end{aligned}
$$

The bound in (16) holds.

Finally, we compute $\nabla_u y_{l+1}^q$. By (10), we have

$$
\begin{aligned}
\nabla_u y_{l+1}^q =& \nabla_u y_l^q + \frac{\mathrm{d}\left[uXX^\top Ax^q\right]}{\mathrm{d}u} + \frac{\mathrm{d}\left[Y_l X^\top Ax^q\right]}{\mathrm{d}u} \\
=& \nabla_u y_l^q + XX^\top Ax^q + (\nabla_u Y_l)X^\top Ax^q.
\end{aligned}
$$

Then, we bound $\left\|\nabla_u y_{l+1}^q\right\|_1$ by

$$
\begin{aligned}
\left\|\nabla_u y_{l+1}^q\right\|_1 =& \left\|\nabla_u y_l^q + XX^\top Ax^q + (\nabla_u Y_l)X^\top Ax^q\right\|_1 \\
\leq& \|\nabla_u y_l^q\|_1 + \left\|XX^\top Ax^q\right\|_1 + \left\|(\nabla_u Y_l)X^\top Ax^q\right\|_1 \\
\leq& \|\nabla_u y_l^q\|_1 + \|X\|_1 \|g(A)\|_1 + \|(\nabla_u Y_l)\|_1 \|g(A)\|_1 \\
\leq& \|\nabla_u y_l^q\|_1 + (1+\gamma)\|g(A)\|_1 + l\zeta\|g(A)\|_1 && \text{(eq. (16))} \\
=& \|\nabla_u y_l^q\|_1 + (1+\gamma)\|\phi_q\|_1 + l\zeta\|\phi_q\|_1 && \text{(eq. (14))} \\
=& \|\nabla_u y_l^q\|_1 + (1+\gamma) + l\zeta
\end{aligned}
$$

Define $\eta \doteq (1+\gamma)$. We prove the bound

$$
\|\nabla_u y_l^q\|_1 \leq \frac{l^2+l}{2}\zeta + l\eta \tag{17}
$$

by induction. We first have the base case $\|\nabla_u y_0^q\|_1 \leq 0$ holds because y_0^q is independent from u and thus $\nabla_u y_0^q = 0$. Now, suppose (17) holds for l. We have

$$
\begin{aligned}
\left\|\nabla_u y_{l+1}^q\right\|_1 &\leq \|\nabla_u y_l^q\|_1 + \eta + l\zeta \\
&\leq \frac{l^2+l}{2}\zeta + l\eta + \eta + l\zeta \\
&\leq \frac{l^2+l}{2}\zeta + (l+1)\zeta + (l+1)\eta \\
&= \frac{l^2+3l+2}{2}\zeta + (l+1)\eta \\
&= \frac{(l+1)^2+(l+1)}{2}\zeta + (l+1)\eta.
\end{aligned}
$$

The bound holds for (17).

Since $\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right) = -y_L^q$, we have

$$
\begin{cases}
\left\|\nabla_A \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1 = \|\nabla_A y_L^q\|_1 \leq \frac{L^2+L}{2}\beta; \\
\left\|\nabla_u \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1 = \|\nabla_u y_L^q\|_1 \leq \frac{L^2+L}{2}\zeta + L\eta.
\end{cases}
$$

We then have bounds

$$
\begin{cases}
\left\|\mathrm{vec}(\nabla_A \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right))\right\|_1 \leq 2n\left(\frac{L^2+L}{2}\beta\right); \\
\left\|\nabla_u^\top \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1 \leq 2n\left(\frac{L^2+L}{2}\zeta + L\eta\right).
\end{cases}
$$

We therefore get

$$
\begin{aligned}
\left\|\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1 =& \left\|\begin{bmatrix}\mathrm{vec}(\nabla_A \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)) \\ \nabla_u^\top \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\end{bmatrix}\right\|_1 \\
=& \left\|\mathrm{vec}(\nabla_A \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right))\right\|_1 + \left\|\nabla_u^\top \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1
\end{aligned}
$$

$$\leq \frac{L^2+L}{2}(2n(\beta+\zeta)) + L(2n\eta).$$

Let $\nu \doteq 2n(\beta+\zeta)$ and $\xi \doteq 2n\eta$. We finally arrive at

$$\left\|\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1 \leq \frac{L^2+L}{2}\nu + L\xi.$$

□

A.5 Theorem 2

Proof. We have

$$\begin{aligned}
&J(\theta_L^{\mathrm{TD}}) \\
\doteq&\left\|\mathbb{E}_{s_q\sim\mu^p, s_q'\sim p(s_q), p\sim\Delta(p), r\sim\Delta(r)}\left[\left(r(s_q)+\gamma \mathrm{TF}_L\left(Z_0';\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right)\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right]\right\|_1 \\
=&\left\|\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\mathbb{E}_{s_q'\sim p(s_q)}\left[\left(r(s_q)+\gamma \mathrm{TF}_L\left(Z_0';\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right)\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right]\right]\right\|_1 \\
=&\left\|\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\mathbb{E}_{s_q'\sim p(s_q)}\left[\left(r(s_q)+\gamma \mathrm{TF}_L\left(Z_0';\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right)\right]\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right]\right\|_1 \\
\leq&\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\left\|\mathbb{E}_{s_q'\sim p(s_q)}\left[\left(r(s_q)+\gamma \mathrm{TF}_L\left(Z_0';\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right)\right]\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1\right] \\
=&\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\left|\mathbb{E}_{s_q'\sim p(s_q)}\left[\left(r(s_q)+\gamma \mathrm{TF}_L\left(Z_0';\theta_L^{\mathrm{TD}}\right)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right)\right]\right|\left\|\nabla_\theta \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1\right] \\
\leq&\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\|r\|_\infty \gamma^L \left\|\nabla_{\theta_L} \mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right\|_1\right] && \text{(Lemma 1)} \\
\leq&\mathbb{E}_{s_q\sim\mu^p, p\sim\Delta(p), r\sim\Delta(r)}\left[\|r\|_\infty \gamma^L\left(\frac{L^2+L}{2}\nu + L\xi\right)\right] && \text{(Lemma 3)} \\
=&\mathbb{E}_{r\sim\Delta(r)}[\|r\|_\infty]\gamma^L\left(\frac{L^2+L}{2}\nu + L\xi\right).
\end{aligned}$$

Let $C_r \doteq \mathbb{E}_{r\sim\Delta(r)}[\|r\|_\infty]$. We have $J(\theta_L^{\mathrm{TD}}) \leq \frac{L^2+L}{2}\nu C_r\gamma^L + L\xi C_r\gamma^L$. Therefore, it holds that

$$\lim_{L\to\infty} J(\theta_L^{\mathrm{TD}}) \leq \lim_{L\to\infty} \frac{L^2+L}{2}\nu C_r\gamma^L + L\xi C_r\gamma^L = 0$$

because the exponential term γ^L dominates the polynomial terms of L. Since $J(\theta_L^{\mathrm{TD}})$ is a norm and thus always nonnegative, we have

$$\lim_{L\to\infty} J(\theta_L^{\mathrm{TD}}) = 0,$$

completing the proof. □

A.6 Lemma 4

Proof. We note that given p, r, s_q, there is no randomness in $\mathrm{TF}_L(Z_0;\theta)$. Therefore, we have

$$\begin{aligned}
&\left|\mathbb{E}_{p,r}\left[G(s_q)-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right]\right| \\
=&\left|\mathbb{E}_{p,r}[G(s_q)]-\mathrm{TF}_L\left(Z_0;\theta_L^{\mathrm{TD}}\right)\right| \\
=&\left|v(s_q)-\langle\phi_q, w_L\rangle\right| \\
=&\left|v(s_q)-\phi_q^\top\sum_{k=0}^{L-1}(\gamma P)^k r\right| && \text{(eq. (6))} \\
=&\left|\phi_q^\top v-\phi_q^\top\left(I-(\gamma P)^L\right)(I-\gamma P)^{-1}r\right|
\end{aligned}$$

$$
\begin{aligned}
&= \left|\phi_q^\top v - \phi_q^\top \left(I - \gamma^L P^L\right) v\right| \\
&= \gamma^L \left|\phi_q^\top P^L v\right| \\
&\leq \gamma^L \|\phi_q\|_1 \left\|P^L v\right\|_\infty && \text{(Hölder's inequality)} \\
&\leq \gamma^L \left\|P^L\right\|_\infty \|v\|_\infty && (\|\phi_q\|_1 = 1) \\
&= \gamma^L \|v\|_\infty. && (\left\|P^L\right\|_\infty = 1)
\end{aligned}
$$

□

A.7 Corollary 1

Proof. (Corollary 1)

$$
\begin{aligned}
& J'(\theta_L^{\text{TD}}) \\
=& \left\|\mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\left(G(s_q) - \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right) \nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right]\right\|_1 \\
=& \left\|\mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\mathbb{E}_{p,r}\left[\left(G(s_q) - \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right) \nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right]\right]\right\|_1 \\
=& \left\|\mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\mathbb{E}_{p,r}\left[G(s_q) - \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right] \nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right]\right\|_1 \\
\leq& \mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\left\|\mathbb{E}_{p,r}\left[G(s_q) - \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right] \nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right\|_1\right] \\
=& \mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\left|\mathbb{E}_{p,r}\left[G(s_q) - \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right]\right| \left\|\nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right\|_1\right] \\
\leq& \mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\|v\|_\infty \gamma^L \left\|\nabla_\theta \text{TF}_L\left(Z_0; \theta_L^{\text{TD}}\right)\right\|_1\right] && \text{(Lemma 4)} \\
\leq& \mathbb{E}_{s_q \sim \mu^p, p \sim \Delta(p), r \sim \Delta(r)} \left[\|v\|_\infty \gamma^L \left(\frac{L^2 + L}{2} \nu + L\xi\right)\right] && \text{(Lemma 3)} \\
=& \mathbb{E}_{p \sim \Delta(p), r \sim \Delta(r)} [\|v\|_\infty] \gamma^L \left(\frac{L^2 + L}{2} \nu + L\xi\right).
\end{aligned}
$$

Let $C_v \doteq \mathbb{E}_{p \sim \Delta(p), r \sim \Delta(r)}[\|v\|_\infty]$, we have $J'(\theta_L^{\text{TD}}) \leq \frac{L^2+L}{2} \nu C_v \gamma^L + L\xi C_v \gamma^L$. Therefore, it holds that

$$
\lim_{L \to \infty} J'(\theta_L^{\text{TD}}) \leq \lim_{L \to \infty} \frac{L^2 + L}{2} \nu C_v \gamma^L + L\xi C_v \gamma^L = 0
$$

because the exponential term γ^L dominates the polynomial terms of L. Since $J'(\theta_L^{\text{TD}})$ is a norm and thus always nonnegative, we have

$$
\lim_{L \to \infty} J'(\theta_L^{\text{TD}}) = 0,
$$

completing the proof. □

B Experiment Details

B.1 Boyan's Chain

We include an illustration of Boyan's chain and its generation method here.

B.2 Compute Resources

We run our experiments in parallel on a single node of a CPU cluster. The node has 150 CPU cores and 150 GB of memory. The wall clock time it takes to finish running the experiments is about 50 minutes for multi-task TD and 15 hours for multi-task MC.

B.3 Hyperparameters and More Implementation Details

We use NumPy [Harris et al., 2020] for data processing and implementing the MRPs. We use PyTorch [Ansel et al., 2024] to create and train our models. For data visualization, we use Matplotlib [Hunter, 2007] to create the plots. We use Draw.io [JGraph, 2021] to make the Boyan's chain figure.

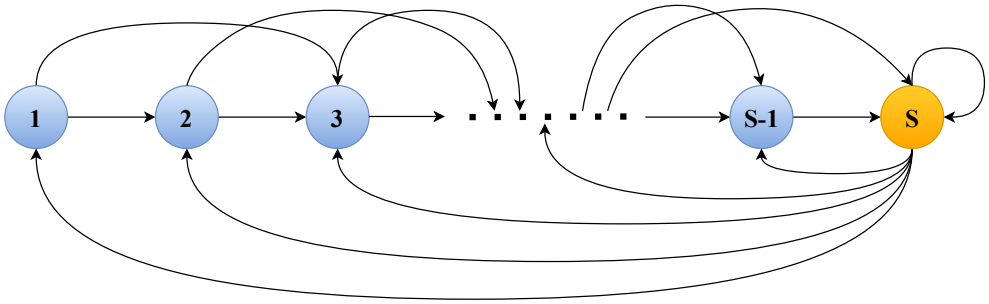


Figure 3: Boyan's chain with S states. Arrows indicate non-zero transition probabilities.

Algorithm 1: Boyan Chain MRP Generation (Adapted from Algorithm 2 of Wang et al. [2025])

Input: state space size $n = |\mathcal{S}|$
$r \sim \text{Uniform}\left[(-1, 1)^n\right]$ `// reward function`
$p \leftarrow 0_{m \times m}$ `// transition function`
for $i = 1, \ldots, n-2$ **do**
  $\epsilon \sim \text{Uniform}\left[(0, 1)\right]$
  $p(i, i+1) \leftarrow \epsilon$
  $p(i, i+2) \leftarrow 1 - \epsilon$
end for
$p(n-1, n) \leftarrow 1$
$z \leftarrow \text{Uniform}\left[(0, 1)^n\right]$
$z \leftarrow z / \sum_s z(s)$
$p(n, 1:n) \leftarrow z$
$p_0 \leftarrow$ `stationary_distribution`(p) `//initial distribution`
Output: MRP (p_0, p, r)

optimizer	Adam [Kingma and Ba, 2015]
learning rate	0.001
weight decay	0.0
batch size	64
# of attention layers	30
# of Boyan's chain states	5
discount factor	0.9
# of Monte Carlo rollout steps	200
# of random seeds	20
# of Boyan's chain tasks for training	20,000
# of validation instances	10
validation context lengths	5, 10, . . . , 100

Table 1: Hyperparameters and more training details.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly define the scope of our work in the abstract and intro. Each claimed contribution is reflected in one or more of the subsequent sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 summarizes the limitations of our work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We state the assumptions in the theory statement or in the context of the theory. We put our complete proof in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We include the important information in Section 6. We put more experimental details in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We still need to clean up the codebase and create instructions to run our code. We will submit our code as part of the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify the experimental details in Section 6 as well as the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We plot the standard errors for our first experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We disclose the compute resources and wall clock times for running the experiments in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We reviewed the Code of Ethics and confirm that our research conforms with all of them.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work is mainly theoretical and fundamental. The perceived societal impacts are minimal.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not use or rely on existing data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the tools we use to implement our experiments and generate our figures in the Appendix.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLM is not involved in the core method development of our paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.