One for All: A Non-Linear Transformer can Enable Cross-Domain Generalization for In-Context Reinforcement Learning

Bowen He* **Juncheng Dong** **Lin Lin** **Xiang Cheng**

Duke University

Abstract

A central challenge in reinforcement learning (RL) is to learn models that generalize beyond the tasks on which they are trained, a goal traditionally pursued through multi-task and meta RL. Recently, transformer architectures have emerged as a promising approach, enabling adaptation to new tasks via in-context learning without explicit parameter updates. From a functional perspective, a transformer can be viewed as a **functional operator** that maps a context to a task-specific function. It is thus fundamental to understand and design this operator to support stronger generalization in RL. In this work, we address this resulting question of generalization from a kernel-based perspective by establishing a connection between non-linear transformers and kernel-based temporal difference learning. By interpreting the transformer as performing regression in a **Reproducing Kernel Hilbert Space** (RKHS), we show that value functions from different domains can be represented using a shared set of weights, provided they lie within the same RKHS. Experiments on multiple MetaWorld domains support this interpretation, demonstrating convergence of the temporal-difference objective.

1 Introduction

Conventional reinforcement learning (RL) methods typically learn task-specific policies and value functions that fail to transfer to new tasks, requiring substantial additional training or even learning from scratch. This has motivated the development of task-agnostic RL methods that support generalization across tasks, commonly referred to as multi-task or meta reinforcement learning. A growing line of work explores the use of transformers for this purpose, leveraging their **in-context learning** capability to implicitly implement adaptation algorithms in their parameters and to condition on task-specific context at inference time. Specifically, in addition to the query state s_{query} at which a decision is required, a transformer also takes as input context information formulated as transitions from an environment $\tau_t \doteq (s_0, a_0, r_0, \cdots, s_{t-1}, a_{t-1}, r_{t-1})$. By conditioning on the context information, the model adapts its behavior for the same s_{query}.

While there has been empirical success in the literature, a principled understanding of the mechanisms underlying in-context RL remains limited, particularly with respect to how and why transformers generalize across tasks, and the conditions under which such generalization emerges. This gap hinders the development of more powerful in-context RL methods that generalize not only across tasks within a single domain, but also across domains themselves. Essentially, a transformer can be viewed as a **functional operator** that generates an implicit function conditioned on the input context. Studying and designing this operator is therefore central to understanding and advancing in-context RL.

*Corresponding author. Email: bowen.he@duke.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In this work, we focus on the policy evaluation setting, where the goal is to estimate the value function of a fixed policy. While policy evaluation alone does not constitute a complete RL procedure—full RL additionally requires policy improvement—it serves as a fundamental subroutine in virtually all modern RL algorithms. Understanding the mechanisms by which in-context models perform policy evaluation is therefore a necessary foundational step toward extending in-context RL to full control. Moreover, focusing on policy evaluation allows us to isolate and rigorously analyze the transformer's ability to learn and generalize value functions, without the additional complexity introduced by policy optimization, leading to clearer theoretical and empirical insights.

One prior work studies this question in a linear setting [26], assuming both a linear Markov Reward Process (MRP) and a linear transformer. Under these assumptions, the authors show that a linear transformer can implement gradient descent on the temporal-difference (TD) loss, thereby fitting state-value functions within the forward pass of the transformer layers. However, this construction relies critically on strong linearity assumptions and therefore applies only to tasks that strictly satisfy these conditions, which limits its applicability in more complex and realistic settings.

Building on this view, we approach the problem from a kernel-based perspective. The attention mechanism can be interpreted as implicitly implementing a kernel function [3], suggesting that nonlinear transformers perform regression in the corresponding Reproducing Kernel Hilbert Space (RKHS) induced by this kernel. This observation establishes a natural connection between nonlinear transformers and kernel-based RL, thereby relaxing the restrictive linearity assumptions on the tasks as well. More importantly, it implies that tasks whose value functions lie in the same RKHS can be addressed by a single transformer, even when they originate from different domains. We validate this hypothesis by applying a transformer with a fixed set of weights across multiple MetaWorld domains, demonstrating convergence of the TD loss. To our knowledge, we provide the first theoretical construction showing that in-context RL with transformers can generalize across domains, offering a foundation for next steps of developing more universal policies with in-context RL.

Contributions.

- **Kernel view of in-context policy evaluation.** We frame in-context *policy evaluation* through the lens of kernel-based reinforcement learning by leveraging the observation that nonlinear attention induces an RKHS kernel, and use this perspective to relate transformer inference to kernelized TD updates (Section 4).
- **Explicit construction of transformer weights.** We give an explicit weight construction showing that a depth-L nonlinear transformer with *fixed* weights can implement L iterations of the kernel TD update purely through its forward pass, using context states as kernel centers (Section 4 and Appendix A).
- **A sufficient condition for cross-domain reuse via a shared RKHS.** Building on the RKHS interpretation, we explain when cross-domain generalization can be expected to hold and when it may fail (Section 4 and Section 5).
- **Empirical support on real tasks.** We instantiate the theoretical construction in practice by applying the transformer to MetaWorld tasks [30], where the empirical results resonate with the theoretical predictions. (Section 6, Appendix B and Appendix C)
- **Transfer analysis.** We evaluate whether a model trained on a single domain can be applied to other domains (Appendix C) and relate its success or failure to kernel-parameter mismatch, consistent with the fixed-kernel limitation discussed in Section 5. We further analyze a practical failure mode arising from discontinuities in the induced value landscape (Appendix D).

2 Related Works

In-context learning. The concept of in-context learning [28] was introduced to describe the phenomenon whereby pretrained large language models adapt their behavior by conditioning on a prompt, referred to as the context. Since its introduction, in-context learning has attracted substantial attention from the research community [2, 5, 8, 9, 27], as it offers a principled mechanism for realizing meta-learning: rather than representing a fixed function, the model implicitly implements an algorithm that adapts its behavior based on the input context.

To further understand the underlying mechanisms, von Oswald et al. [25] consider a linear setting and show that linear transformers can implement a family of gradient-descent algorithms, implicitly optimizing a function in their forward pass. This weight construction was later theoretically supported by subsequent work [1, 31, 16], which shows that it closely matches the global optimum of the in-context loss for a one-layer transformer and that training such models converges to this optimum in polynomial time.

While the above works focus on linear transformers, another line of research interprets the attention mechanism as a kernel operation and designs attention computations based on different kernel choices [24, 4, 6, 18]. Building on this perspective, Cheng et al. [3] further demonstrates that nonlinear transformers can perform kernel regression with induced kernel functions, providing a principled explanation of their behavior.

In-context reinforcement learning. In-context reinforcement learning (RL) introduces additional complexity, as reinforcement learning algorithms typically involve dynamic programming procedures for policy evaluation and policy improvement [21]. To isolate this complexity, prior empirical work pretrains transformers in a supervised manner for decision-making tasks [12, 13], demonstrating that such pretrained models can perform effectively on tasks not explicitly encountered during training. Building on this idea, subsequent methods modify the structure of the context information to provide more informative trajectories, enabling transformers to implement policy improvement within their forward pass [20, 10, 15]. Tarasov et al. [22] further show that combining the in-context RL objective with a Q-learning objective yields direct gains.

On the theoretical side, recent work has begun to formalize the algorithmic capabilities of transformers in in-context RL. Lin et al. [14] provide a provable explanation for supervised pretraining-based approaches, showing that transformers can implement classical reinforcement learning algorithms such as UCB and Thompson sampling through their forward pass. Complementarily, Wang et al. [26] study policy evaluation in the linear setting, demonstrating that transformers can implicitly implement TD algorithms in their learned weights. As in-context RL is a rapidly evolving area encompassing a wide range of paradigms and perspectives, we refer readers to the survey by Moeini et al. [17] for a more comprehensive overview.

Toward the goal of realizing a universal policy through in-context RL, Kirsch et al. [11] provide initial empirical evidence that a transformer pretrained on tasks from a single domain, specifically Ant in their experiments, can perform in-context adaptation to other domains such as Reacher, HalfCheetah, and CartPole. These results support the hypothesis that a transformer with a fixed set of weights can learn in context across multiple domains; however, the work does not offer a formal justification for this behavior. In this work, we take a first step toward providing such an explanation, shedding light on the development of more general and transferable policy models.

Kernel-based reinforcement learning. Kernel-based RL has a long history as a means of replacing the linear function approximation for value estimation, enabling flexible state representations and theoretical guarantees under certain assumptions [19, 7]. A body of work studies kernelized TD learning for policy evaluation, including both batch and online variants, which embed value functions in a RKHS to achieve consistent estimation in continuous state spaces [29, 23]. These methods provide principled alternatives to parametric function approximation by explicitly introducing nonlinear function classes for state-value estimation.

3 Settings

Reinforcement Learning. We formulate decision making as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, p_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{T}$ the transition dynamics, R the reward function, p_0 the initial state distribution, and γ the discount factor [21]. In continuous setting, $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{S}|}$ and $\mathcal{A} \subseteq \mathbb{R}^{|\mathcal{A}|}$, with $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$. For a policy π, the state-value function is

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right],$$

where the expectation is taken over both $\mathcal{T}$ and π.

Fixing π reduces the MDP to a Markov Reward Process (MRP) $(\mathcal{S}, \mathcal{T}^\pi, R^\pi, p_0, \gamma)$, where

$$\mathcal{T}^\pi(s' \mid s) = \int \pi(a \mid s)\mathcal{T}(s' \mid s, a)\, da, \qquad R^\pi(s) = \int \pi(a \mid s)R(s, a)\, da.$$

Correspondingly, transitions up to time step t in the MDP, $(s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1})$, is reduced to $(s_0, r_0, \ldots, s_{t-1}, r_{t-1})$, and the state-value function can be written as

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^\pi(s_t) \,\middle|\, s_0 = s\right],$$

where the expectation is taken only with respect to the transition dynamics $\mathcal{T}^\pi$. The goal of policy evaluation is therefore to compute the state-value function $V(s)$ with interaction trajectories $(s_0, r_0, \ldots, s_{t-1}, r_{t-1})$.

Linear policy evaluation approximates the value function using a linear function of features,

$$V(s) \approx w^\intercal \phi(s),$$

where $\phi(s)$ is a predefined or learned feature map that embeds a state into the feature space. Kernel-based policy evaluation instead models the value function in an RKHS $\mathcal{H}_\kappa$ induced by a kernel κ:

$$V(s) = \langle V, \kappa(s, \cdot)\rangle_{\mathcal{H}_\kappa} \approx \sum_{i=1}^{n} \alpha_i \kappa(s^{(i)}, s).$$

Given the current iterate $V^{(m)}$, we define the empirical TD error as

$$b_t^{(m)} = r_t + \gamma V^{(m)}(s_{t+1}) - V^{(m)}(s_t).$$

The kernelized semi-gradient TD update is thus

$$V^{(m+1)}(\cdot) = V^{(m)}(\cdot) + \alpha_m \sum_{t=0}^{n-1} b_t^{(m)}\, \kappa(s_t, \cdot),$$

which treats the bootstrapped target $r_t + \gamma V^{(m)}(s_{t+1})$ as fixed within each update. Thus, this is the RKHS analogue of classical semi-gradient TD, rather than exact gradient descent on the full squared Bellman residual.

We show that a nonlinear transformer can implement this kernelized TD update in context: each layer performs one TD-style update over the in-context transitions, and the multi-layer forward pass carries out iterative RKHS-based policy evaluation.

Transformer. We define attention as

$$\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z) \doteq VZ \cdot M \cdot \tilde{h}(KZ, QZ),$$

where $Z \in \mathbb{R}^{(2d+1)\times(n+1)}$ is the input, K, Q, V are projection matrices, and

$$\tilde{h} : \mathbb{R}^{(2d+1)\times(n+1)} \times \mathbb{R}^{(2d+1)\times(n+1)} \to \mathbb{R}^{(n+1)\times(n+1)}$$

is the nonlinear activation used in attention. The matrix M is a masking given by

$$M \doteq \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & 0 \end{bmatrix}, \tag{1}$$

which prevents information flow from the query column to the context during attention.

Correspondingly, a transformer layer is defined via a residual update,

$$Z_{l+1} \doteq Z_l + \texttt{Attn}^{\tilde{h}}_{K_l,Q_l,V_l}(Z_l), \tag{2}$$

and its multi-head variant is given by

$$Z_{l+1} \doteq Z_l + \sum_{i=1}^{H} \texttt{Attn}^{\tilde{h}}_{K_l^i,Q_l^i,V_l^i}(Z_l). \tag{3}$$

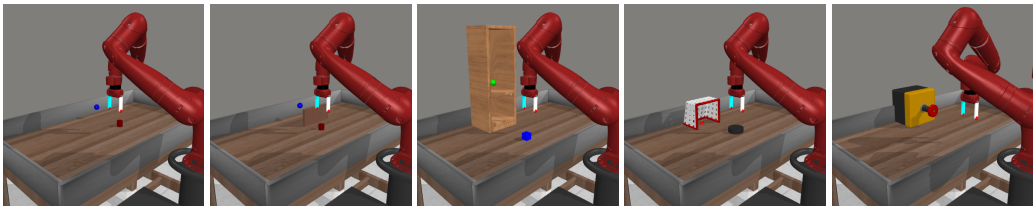


Figure 1: Task domains from MetaWorld. From left to right: Pick-Place, Pick-Place-Wall, Shelf-Place, Plate-Slide, and Button-Press.

For a k-layer transformer, either 2 or 3 is applied iteratively to update the input matrix layer by layer. We denote the output matrices Z_l of the transformer as the output

$$\mathrm{TF}_l^\theta(Z_0) \doteq Z_l, \quad l \in \{1, \ldots, k\},$$

where θ denotes the collection of parameters of the transformer.

Nonlinear Activation. Following Cheng et al. [3], a nonlinear attention activation can be interpreted as inducing a kernel over input vectors. With linear activation,

$$[\tilde{h}(U, V)]_{i,j} \doteq U_i^\intercal V_j,$$

the induced kernel is linear:

$$\kappa(x, y) = x^\intercal y.$$

With exponential activation,

$$[\tilde{h}(U, V)]_{i,j} \doteq \exp\left(\frac{U_i^\intercal V_j}{\delta}\right), \tag{4}$$

the induced kernel becomes

$$\kappa(x, y) = \exp\left(\frac{x^\intercal y}{\delta}\right), \tag{5}$$

where δ is a temperature parameter. This kernel perspective allows us to view nonlinear transformers with fixed weights as performing kernel regression in their forward pass. Building on this interpretation, we connect nonlinear transformers to kernel-based policy evaluation and show how a single set of weights can operate across multiple domains.

4 The In-Context Kernel TD Update

Notation. We use i, j to denote the *trajectory* time index, and k, ℓ to denote the iteration index of the TD algorithm. Let $s^{(i)} \in \mathbb{R}^d$ denote the state vector at trajectory index i and let $r^{(i)} \in \mathbb{R}$ denote the reward at time i. Let $v_k^{(i)}$ denote the value of state $s^{(i)}$ after k iterations of the TD update. Define the TD residual

$$b_k^{(i)} := r^{(i)} + \gamma v_k^{(i+1)} - v_k^{(i)}.$$

We assume the kernel κ is symmetric, i.e. $\kappa(x, y) = \kappa(y, x)$.

Kernel TD update. One step of the kernel TD update is

$$v_{k+1}^{(i)} = v_k^{(i)} + \alpha_k \sum_{j=0}^{n-1} b_k^{(j)} \kappa(s^{(j)}, s^{(i)}). \tag{6}$$

Equivalent update for $b_k^{(i)}$. Using the definition of $b_k^{(i)}$ and (6), we obtain

$$\begin{aligned} b_{k+1}^{(i)} &= b_k^{(i)} + \gamma\left(v_{k+1}^{(i+1)} - v_k^{(i+1)}\right) - \left(v_{k+1}^{(i)} - v_k^{(i)}\right) \\ &= b_k^{(i)} + \underbrace{\gamma\alpha_k \sum_{j=0}^{n-1} b_k^{(j)} \kappa(s^{(j)}, s^{(i+1)})}_{\texttt{Attn}^2} - \underbrace{\alpha_k \sum_{j=0}^{n-1} b_k^{(j)} \kappa(s^{(j)}, s^{(i)})}_{\texttt{Attn}^1}. \end{aligned} \tag{7}$$

Initialization. We initialize $v_0^{(i)} = 0$ for all i. Consequently, $b_0^{(i)} = r^{(i)}$ for $i = 0, \ldots, n-1$. For the query column we set $b_0^{(n)} = 0$. We will construct a self-attention layer with two heads (denoted $\texttt{Attn}^1$ and $\texttt{Attn}^2$) which exactly compute the two underbraced expressions in (7).

4.1 Main Theoretical Result

Let $Z_\ell \in \mathbb{R}^{(2d+1)\times(n+1)}$ denote the output after ℓ Transformer layers. We will show that a single fixed set of Transformer weights can implement ℓ steps of the TD iteration (6) *in-context*: the weights are universal, while the input matrix Z_0 encodes the particular context states and the query state.

Let $s^{(\texttt{pad})}$ denote a fixed padding state (can be taken to be 0). For each TD iteration index k, recall that $v_k^{(i)}$ denotes the value of state $s^{(i)}$ after k iterations of TD; in particular, $v_k^{(\texttt{pad})}$ denotes the value of the padding state $s^{(\texttt{pad})}$ after k TD iterations. For each $\ell \in \{0, 1, \ldots, L\}$, define

$$Z_\ell \;=\; \begin{bmatrix} s^{(0)} & \cdots & s^{(n-1)} & s^{(n)} \\ s^{(1)} & \cdots & s^{(n)} & s^{(\texttt{pad})} \\ b_\ell^{(0)} & \cdots & b_\ell^{(n-1)} & -v_\ell^{(n)} + \gamma v_\ell^{(\texttt{pad})} \end{bmatrix}. \tag{8}$$

Here the scalars $\{b_\ell^{(i)}\}_{i=0}^{n-1}$ and $v_\ell^{(n)}$ are the iterates produced by the ℓ-step TD update (6) (with kernel κ defined in (5)) when applied to the trajectory $(s^{(0)}, \ldots, s^{(n)})$.

For ease of discussion, we explicitly specify the initialization

$$Z_0 \;=\; \begin{bmatrix} s^{(0)} & \cdots & s^{(n-1)} & s^{(n)} \\ s^{(1)} & \cdots & s^{(n)} & s^{(\texttt{pad})} \\ r^{(0)} & \cdots & r^{(n-1)} & 0 \end{bmatrix}. \tag{9}$$

Theorem 4.1 (In-context implementation of L steps of TD)**.** *Fix n and let $s^{(\texttt{pad})}$ be a padding state. Consider the TD iteration (6) with kernel κ defined in (5). There exist Transformer weights*

$$\theta \;=\; \Big\{(K_\ell^j, Q_\ell^j, V_\ell^j) : j \in \{0,1\},\ \ell \in \{1, \ldots, L\}\Big\},$$

independent of the particular trajectory, such that the following holds: for any trajectory $(s^{(0)}, r^{(0)}, \ldots, s^{(n-1)}, r^{(n-1)}, s^{(n)})$, if Z_0 is defined as in (9), then

$$\texttt{TF}_\ell^\theta(Z_0) \;=\; Z_\ell \qquad \textit{for all } \ell \in \{1, \ldots, L\},$$

where Z_ℓ is as defined in (8). In particular, the last row of Z_ℓ contains the ℓth iteration TD values for each state.

Discussion. The transformer uses states from the context transitions as kernel centers for the TD update and directly evaluates the query state $s^{(n)}$ to obtain its value. The value can be read out directly as the negative of the bottom-right entry of the matrix Z_l. In this sense, the transformer is explicitly equipped with a TD algorithm for estimating state values and primarily requires transition data from a new MRP to perform this computation. However, the current procedure introduces a constant bias that is independent of the query state; we defer a discussion of this bias to Section 5. Nevertheless, this formulation establishes a principled connection between non-linear transformers and kernelized reinforcement learning, thereby enabling analysis through the lens of kernel-based RL. We provide the ***explicit constructions of the the two heads*** in appendix A with additional details to analyze the computation.

Proof of Theorem 4.1. We explicitly construct the weights θ by specifying, for each layer $\ell \in \{0, 1, \ldots, L-1\}$, a two-head attention block whose first head implements the term $\texttt{Attn}^1$ and whose second head implements the term $\texttt{Attn}^2$ in (7). Concretely, in layer $\ell + 1$ we set

$(K_{\ell+1}^1, Q_{\ell+1}^1, V_{\ell+1}^1)$ to be the matrices in Lemma A.1,

$(K_{\ell+1}^2, Q_{\ell+1}^2, V_{\ell+1}^2)$ to be the matrices in Lemma A.2.

We prove by induction on ℓ that the layer outputs satisfy $\texttt{TF}_\ell^\theta(Z_0) = Z_\ell$, where Z_ℓ is defined in (8).

Base case. By construction, the Transformer input is initialized as Z_0 in (9). This matches (8) at $\ell = 0$ since $v_0^{(i)} = 0$ for all i and thus $b_0^{(i)} = r^{(i)}$ for $i = 0, \ldots, n-1$, with the query-column residual set to $b_0^{(n)} = 0$.

Inductive step. Assume for some $\ell \in \{0, 1, \ldots, L-1\}$ that $\mathtt{TF}_\ell^\theta(Z_0) = Z_\ell$. Apply the $(\ell+1)$-st Transformer layer. By Lemma A.1, head 1 produces an update matrix whose only nonzero entries lie in the last row and equal

$$-\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)} \kappa(s^{(j)}, s^{(i)}) \quad \text{in column } i.$$

By Lemma A.2, head 2 produces an update matrix whose only nonzero entries lie in the last row and equal

$$\gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)} \kappa(s^{(j)}, s^{(i+1)}) \quad \text{in column } i \in \{0, \ldots, n-1\}, \text{and}$$

$$\gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)} \kappa(s^{(j)}, s^{(\mathtt{pad})}) \quad \text{in column } n.$$

Summing the two heads and adding the result to the current last row (as per the Transformer layer definition) yields, for each $i \in \{0, \ldots, n-1\}$,

$$b_{\ell+1}^{(i)} = b_\ell^{(i)} + \alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)} \Big(\gamma\, \kappa(s^{(j)}, s^{(i+1)}) - \kappa(s^{(j)}, s^{(i)}) \Big).$$

which is exactly the TD residual update (7). The first two block-rows are unchanged by our choice of V in both heads, hence remain $(s^{(0)}, \ldots, s^{(n)})$ and $(s^{(1)}, \ldots, s^{(n)}, s^{(\mathtt{pad})})$. Finally, the query column update equals the same expression with $s^{(i+1)}$ replaced by $s^{(\mathtt{pad})}$, giving precisely the form $v_{\ell+1}^{(n)} + \gamma v_{\ell+1}^{(\mathtt{pad})}$ in (8).

Therefore the layer output equals $Z_{\ell+1}$, completing the inductive step.

By induction, $\mathtt{TF}_\ell^\theta(Z_0) = Z_\ell$ for all $\ell \in \{1, \ldots, L\}$. This conclude the proof. □

5 Limitations

While our construction establishes a principled connection between transformers and kernel-based TD learning, several important limitations remain that warrant discussion and suggest promising directions for future work.

Constant offset in value estimation. The current construction introduces a state-independent offset to all value estimates, specifically

$$V(s^{(n)}) = v_\ell^{(n)} - \gamma v_\ell^{(\mathtt{pad})}.$$

This offset implies that the absolute value estimates contain a constant bias that does not vanish even when the underlying value function is perfectly represented within the RKHS. However, this limitation has restricted practical impact for two key reasons. First, in policy improvement procedures such as policy gradient methods or advantage-based algorithms, only *relative* value comparisons between states or state-action pairs matter. The constant offset cancels when computing advantages or comparing action values, leaving policy learning unaffected. Second, when computing TD errors during training, the offset appears in both $V(s_t)$ and $V(s_{t+1})$, causing it to cancel in the TD error. This ensures that the TD loss converges to zero during training despite the absolute value bias, as we empirically demonstrate in Section 6 about limitations.

Nevertheless, addressing this offset through architectural modifications—such as introducing additional normalization layers or learnable bias terms—represents a natural avenue for improving the construction's accuracy for applications requiring absolute value estimates.

Fixed kernel parameters. A more fundamental limitation is that the kernel function parameters, such as the temperature δ in the exponential kernel $\kappa(x, y) = \exp(x^\intercal y/\delta)$, remain fixed throughout

the forward pass and are not adapted based on the input context. This restricts the transformer's ability to handle tasks whose value functions require fundamentally different kernel characteristics. For instance, consider two tasks: one requiring smooth, slowly-varying value functions (suggesting a large kernel bandwidth) and another exhibiting sharp, localized value structure (requiring a small bandwidth). A single fixed kernel cannot optimally represent both.

From the perspective of our RKHS interpretation, this limitation can be understood as follows. Each choice of kernel κ induces a distinct RKHS $\mathcal{H}_\kappa$ with its own native norm and function characteristics. Our construction guarantees accurate value function approximation only when $V^\star \in \mathcal{H}_\kappa$, i.e., when the true value function lies within the RKHS induced by the chosen kernel. If two tasks induce value functions $V_1^\star$ and $V_2^\star$ that lie in different RKHSs, then no single fixed kernel can represent both functions well.

This naturally raises the question of whether kernel parameters can be adapted in context, dynamically adjusting the induced RKHS based on contextual and task-specific characteristics, or whether it is possible to learn the kernel itself in context. We leave the investigation of these directions for future work.

Despite these limitations, our construction and analysis provide a justification for why transformers with fixed weights can perform policy evaluation across multiple domains, laying the groundwork for developing more sophisticated in-context RL methods with principled generalization guarantees.

6 Experiments

MetaWorld [30] provides an ideal testbed for analyzing multi-task RL methods. It features a diverse set of environments, each consisting of 50 task variations with different goal configurations. These environments can be viewed as distinct domains, as they require qualitatively different manipulation behaviors from the robot arm, posing a significant challenge for generalization. In addition, Meta-World provides implementations of near-optimal policies for each domain. Accordingly, for each task within a domain, we construct the corresponding MRPs by injecting noise of different levels into the actions generated by the associated policy.

6.1 Implementation

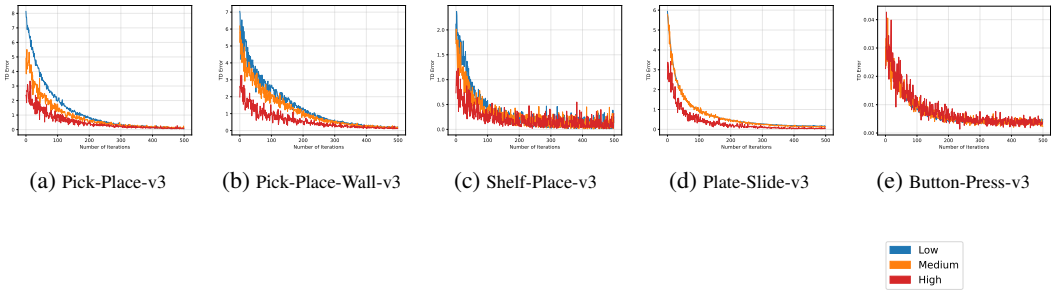
Our experiments are designed to validate the theoretical in-context TD mechanism rather than to train a generic transformer end-to-end. Accordingly, we instantiate the attention weights exactly as constructed in Appendix A. We stack multiple transformer layers, each constructed to perform policy evaluation. During training, we introduce a learnable scalar parameter α and a constant n to modulate the residual update, given by

$$Z_{l+1} \doteq Z_l + \frac{\alpha}{n} \sum_{i=1}^{2} \texttt{Attn}^{\tilde{h}}_{K_l^i, Q_l^i, V_l^i}(Z_l).$$

Intuitively, α corresponds to the effective learning rate of the kernel regression implicitly implemented in the forward pass and therefore must be learned. The constant n denotes the context length and is introduced to stabilize training. In our experiments, we adopt the exponential activation and the corresponding exponential kernel in the transformer layers, as this choice yields an exact equivalence between the attention computation and the induced kernel. The softmax activation is closely related to the exponential kernel, differing only by an additional normalization factor in the underlying kernel function, as noted by [3]. The choice between exponential and softmax activations effectively determines the induced RKHS.

6.2 One-Domain Generalization

We select the following domains to evaluate our transformer: **Pick-Place-v3**, **Pick-Place-Wall-v3**, **Shelf-Place-v3**, **Plate-Slide-v3**, and **Button-Press-v3**. Figure 1 illustrate the different domains. They span a range of difficulty levels and return scales, enabling us to verify the hypothesis underlying the proposed kernel-based perspective. Intuitively, tasks within the same domain are expected to exhibit similar state-value function landscapes, whereas tasks from different domains should induce more distinct function classes. If a transformer with a single set of weights can successfully model value



(a) Pick-Place-v3 (b) Pick-Place-Wall-v3 (c) Shelf-Place-v3 (d) Plate-Slide-v3 (e) Button-Press-v3

Figure 2: Cross-task TD errors for multiple model checkpoints during training.

functions across such heterogeneous domains, it would provide strong evidence for cross-domain generalization.

For each task within a domain, we inject noise at **high**, **medium**, and **low** levels to construct distinct MRPs. Table 1 from appendix E.1 reports the noise std and the estimated initial state values under different noise levels, demonstrating a coverage of the value range. To train the proposed transformer, we keep the model weights fixed and update only the scaling coefficients of the residual connections between layers. During training, we expose the transformer only to tasks with **high** noise levels for each domain, while evaluating its TD error on tasks with **medium** and **low** noise levels. This setup allows us to assess the in-context learning capability of the model across diverse MRPs, even within the same domain. Since each domain contains **50** tasks, evaluating three noise levels results in a total of **150** MRPs. To provide a comprehensive evaluation, at each evaluation point we sample **5** tasks from the MRPs at different noise levels and report the mean TD error.

Figure 2 reports the temporal-difference losses over the entire training process for each domain. *We restrict value computation to early episode states for practical reasons, as the induced value function is highly discontinuous over the full state space and thus challenging to estimate accurately.* A detailed discussion of this issue is provided in Appendix D. Nevertheless, the results show consistent decreases in TD loss across all domains, providing evidence that a transformer with a single set of weights can be effectively applied across domains and achieve convergence in terms of TD loss.

Linear Transformer Fails. To provide additional insight, we report in Appendix B the learning curves of the non-linear transformer alongside those of the linear transformer from Wang et al. [26]. As expected, the linear transformer performs poorly on MetaWorld tasks, since it can only represent linear functions, which is insufficient to capture the underlying value-function structure.

6.3 Cross-Domain Generalization

Another important question is whether the transformer can generalize across different domains, a central question directly reflected in the title of our work. As discussed in Section 5, we anticipate that such generalization is possible as long as the underlying functions can be well approximated within a single RKHS. To investigate this empirically, we present results in Appendix C, where models trained on one domain at different stages of training are applied to multiple domains. We observe that the models generalize well across Pick-Place-v3, Pick-Place-Wall-v3, Shelf-Place-v3, and Plate-Slide-v3, but fail to transfer to Button-Press-v3. This behavior is consistent with the hyperparameter analysis in Appendix E.2, where the four domains share a common kernel parameter, while Button-Press-v3 requires a substantially different setting.

It empirically supports our hypothesis that a single model can generalize not only across tasks within the same domain, but also across different domains.

6.4 Synthetic Environment

We design a synthetic MRP whose state value function lies in an RKHS. The state space of the MRP is two-dimensional, which allows us to directly visualize the learned and ground-truth value functions using three-dimensional surface plots. Figure 3 compares the model-approximated state value function with the ground-truth state value function. Specifically, to construct the transformer



(a) Model approximated state value function. (b) Ground truth state value function.

Figure 3: Comparison between the model-approximated and ground-truth state value functions.

input Z_0, we uniformly sample 32 states from the state space as $s_0, \ldots, s_{n-1}$, together with their corresponding next states and rewards as context. We then fix the context and vary the query state across the state space to obtain the predicted state values. We use 30 transformer layers, consistent with the setting used in the MetaWorld experiments.

As illustrated in Figure 3, the transformer accurately captures the overall shape of the state value function. Although the value ranges differ due to the constant bias term discussed in Section 5, the relative value comparisons are captured well. This is reflected in the close alignment between the shapes of the model-approximated and ground-truth value functions. Numerically, after accounting for the constant bias in the state-value estimates, the model achieves a mean state-value error as low as 0.00034. We provide additional visualizations, ablation studies, and environment details in Appendix F. These results further support our hypothesis.

7 Discussion

This work provides a first theoretical construction that transformers with non-linear activations can generalize not only across tasks within a single domain, but also across domains in in-context reinforcement learning. By adopting a kernel-based reinforcement learning perspective, we establish a principled connection between non-linear transformers and kernel-based temporal-difference learning, showing that a single transformer with fixed weights can implement policy evaluation across multiple domains when the induced value functions lie within a shared RKHS.

Our analysis assumes a fixed RKHS induced by the transformer's attention mechanism. Under this restriction, performance degrades when the underlying value function lies outside the induced function space. Importantly, this limitation is not merely empirical but follows directly from the RKHS interpretation: tasks whose value functions require substantially different kernel characteristics cannot be simultaneously represented by a single fixed kernel. This perspective provides a principled explanation for the observed empirical behavior, including cases where cross-domain transfer fails.

These observations naturally motivate future research directions, such as enabling in-context adaptation of kernel parameters or learning kernel representations directly from context, thereby allowing the induced function space to adjust to task-specific structure.

Finally, our focus on policy evaluation enables a precise theoretical analysis but leaves open questions regarding policy improvement and control. Extending the kernel-based interpretation developed here to these settings, as well as to architectures with adaptive kernel structure, represents a natural direction for future work. We view the present work as a foundational step toward understanding how such capabilities might emerge in in-context reinforcement learning.

Acknowledgments and Disclosure of Funding

B. He and L. Lin were supported by the Duke University Center for AIDS Research (CFAR), an NIH-funded program (P30 AI064518-22), and by NIH grant P01 AI129859-09 for research on in-context learning, in-context reinforcement learning, and their applications to health, and other domains. We thank the Department of Computer Science at Duke University for computational resources. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

[1] K. Ahn, X. Cheng, H. Daneshmand, and S. Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8ed3d610ea4b68e7afb30ea7d01422c6-Abstract-Conference.html.

[2] E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

[3] X. Cheng, Y. Chen, and S. Sra. Transformers implement functional gradient descent to learn non-linear functions in context. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 8002–8037. PMLR, 2024. URL https://proceedings.mlr.press/v235/cheng24a.html.

[4] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller. Rethinking attention with Performers. In *The Ninth International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

[5] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, and F. Wei. Why can GPT learn in-context? Language models secretly perform gradient descent as meta-optimizers. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.247. URL https://aclanthology.org/2023.findings-acl.247.

[6] A. El-Nouby, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek, and H. Jegou. XCiT: Cross-covariance image transformers. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/a655fbe4b8d7439994aa37ddad80de56-Abstract.html.

[7] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, ICML '05, pages 201–208. Association for Computing Machinery, 2005. doi: 10.1145/1102351.1102377. URL https://doi.org/10.1145/1102351.1102377.

[8] S. Garg, D. Tsipras, P. Liang, and G. Valiant. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/c529dba08a146ea8d6cf715ae8930cbe-Abstract-Conference.html.

[9] M. Hahn and N. Goyal. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*, 2023. URL https://arxiv.org/abs/2303.07971.

[10] S. Huang, J. Hu, H. Chen, L. Sun, and B. Yang. In-context decision transformer: Reinforcement learning via hierarchical chain-of-thought. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 19871–19885. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24j.html.

[11] L. Kirsch, J. Harrison, C. D. Freeman, J. Sohl-Dickstein, and J. Schmidhuber. Towards general-purpose in-context learning agents. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=zDTqQVGgzH.

[12] M. Laskin, L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald, D. Strouse, S. Hansen, A. Filos, E. Brooks, M. Gazeau, H. Sahni, S. Singh, and V. Mnih. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=hy0a5MMPUv.

[13] J. N. Lee, A. Xie, A. Pacchiano, Y. Chandak, C. Finn, O. Nachum, and E. Brunskill. Supervised pretraining can learn in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/8644b61a9bc87bf7844750a015feb600-Abstract-Conference.html.

[14] L. Lin, Y. Bai, and S. Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=CqMhoTVgHX.

[15] H. Liu and P. Abbeel. Emergent agentic transformer from chain of hindsight experience. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 21362–21374. PMLR, 2023. URL https://proceedings.mlr.press/v202/liu23a.html.

[16] A. Mahankali, T. B. Hashimoto, and T. Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/87f476af4053961667c2c08e9f4b850e-Abstract-Conference.html.

[17] A. Moeini, J. Wang, J. Beck, E. Blaser, S. Whiteson, R. Chandra, and S. Zhang. A survey of in-context reinforcement learning, 2025. URL https://arxiv.org/abs/2502.07978.

[18] T. Nguyen, T. M. Nguyen, D. D. Le, D. K. Nguyen, V.-A. Tran, R. G. Baraniuk, N. Ho, and S. J. Osher. Improving transformers with probabilistic attention keys. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 16595–16621. PMLR, 2022. URL https://proceedings.mlr.press/v162/nguyen22c.html.

[19] D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2): 161–178, Nov. 2002. ISSN 1573-0565. doi: 10.1023/A:1017928328829. URL https://doi.org/10.1023/A:1017928328829.

[20] L. X. Shi, Y. Jiang, J. Grigsby, L. Fan, and Y. Zhu. Cross-episodic curriculum for transformer agents. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/001608167bb652337af5df0129aeaabd-Abstract.html.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. MIT Press, Cambridge, MA, 1998. ISBN 9780262193986. URL https://books.google.com/books?id=CAFR6IBF4xYC.

[22] D. Tarasov, A. Nikulin, I. Zisman, A. Klepach, A. Polubarov, N. Lyubaykin, A. Derevyagin, I. Kiselev, and V. Kurenkov. Yes, Q-learning helps offline in-context RL. *Reinforcement Learning Journal*, 7, 2026. URL https://rlj.cs.umass.edu/2026/papers/Paper3.html.

[23] G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1017–1024, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553504. URL https://doi.org/10.1145/1553374.1553504.

[24] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443. URL https://aclanthology.org/D19-1443.

[25] J. von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International*

Conference on Machine Learning (ICML), volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR, 2023. URL https://proceedings.mlr.press/v202/von-oswald23a.html.

[26] J. Wang, E. Blaser, H. Daneshmand, and S. Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2e3435554b430bd8fe92a60c509929a0-Abstract-Conference.html.

[27] X. Wang, W. Zhu, M. Saxon, M. Steyvers, and W. Y. Wang. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 15614–15638, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3255a7554605a88800f4e120b3a929e1-Abstract-Conference.html.

[28] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit Bayesian inference. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=RdJVFCHjUMI.

[29] X. Xu, D. Hu, and X. Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, July 2007. ISSN 1045-9227. doi: 10.1109/TNN.2007.899161. URL https://doi.org/10.1109/TNN.2007.899161.

[30] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning (CoRL 2019)*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

[31] R. Zhang, S. Frei, and P. L. Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024. URL https://jmlr.org/papers/v25/23-1042.html.

A Construction Lemmas

Lemma A.1 (Construction for $\texttt{Attn}^1$)**.** *Recall $\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z) = VZM\tilde{h}(KZ, QZ)$, with mask M as defined in (1). Let*

$$K = \begin{bmatrix} \boldsymbol{I}_d & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} \boldsymbol{I}_d & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & 0 \end{bmatrix},$$

$$V = \begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & -\alpha_\ell \end{bmatrix}.$$

Assume $\tilde{h}$ induces the kernel κ on the first d coordinates, i.e. $[\tilde{h}(KZ_\ell, QZ_\ell)]_{j,i} = \kappa(s^{(j)}, s^{(i)})$. Then for any Z_ℓ of the form (8),

$$\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z_\ell) = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ y_\ell^{(0)} & \cdots & y_\ell^{(n-1)} & y_\ell^{(n)} \end{bmatrix},$$

where for each $i \in \{0, 1, \ldots, n\}$,

$$y_\ell^{(i)} \;:=\; -\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(i)}).$$

Proof. Write $Z_\ell = \begin{bmatrix} Z_\ell^{(1)} \\ Z_\ell^{(2)} \\ Z_\ell^{(3)} \end{bmatrix}$ where $Z_\ell^{(1)} \in \mathbb{R}^{d\times(n+1)}$ is the first block-row, $Z_\ell^{(2)} \in \mathbb{R}^{d\times(n+1)}$ is the second block-row, and $Z_\ell^{(3)} \in \mathbb{R}^{1\times(n+1)}$ is the last row.

Step 1: KZ_ℓ and QZ_ℓ. By the definitions of K and Q, we have

$$KZ_\ell = \begin{bmatrix} Z_\ell^{(1)} \\ \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{1\times(n+1)} \end{bmatrix}, \qquad QZ_\ell = \begin{bmatrix} Z_\ell^{(1)} \\ \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{1\times(n+1)} \end{bmatrix}.$$

Hence, by the assumption on $\tilde{h}$ in (4) and (5),

$$\left[\tilde{h}(KZ_\ell, QZ_\ell)\right]_{j,i} = \kappa(s^{(j)}, s^{(i)}).$$

Step 2: VZ_ℓ. By the definition of V, VZ_ℓ is zero in its first $2d$ rows and equals $-\alpha_\ell Z_\ell^{(3)}$ in its last row:

$$VZ_\ell = \begin{bmatrix} \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{d\times(n+1)} \\ -\alpha_\ell Z_\ell^{(3)} \end{bmatrix}.$$

Step 3: masking by M. Since $M = \begin{bmatrix} \boldsymbol{I}_n & 0 \\ 0 & 0 \end{bmatrix}$, multiplying on the right zeros out the last (query) column, so

$$VZ_\ell\, M = \begin{bmatrix} \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{d\times(n+1)} \\ -\alpha_\ell\,[b_\ell^{(0)}, \ldots, b_\ell^{(n-1)}, 0] \end{bmatrix}.$$

Step 4: multiply by affinity matrix. Therefore

$$\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z_\ell) = (VZ_\ell\, M)\, H$$

has zero in its first $2d$ rows, and for each $i \in \{0, \ldots, n\}$ its last-row entry is

$$-\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, H_{j,i} = -\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(i)}) \;=\; y_\ell^{(i)}.$$

This concludes the proof. □

Lemma A.2 (Construction for $\texttt{Attn}^2$)**.** *Recall $\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z) = VZ\, M\, \tilde{h}(KZ, QZ)$, with mask M as defined in (1). Let*

$$K = \begin{bmatrix} \boldsymbol{I}_d & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} \mathbf{0}_{d\times d} & \boldsymbol{I}_d & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & 0 \end{bmatrix},$$

$$V = \begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times d} & \gamma\alpha_\ell \end{bmatrix}.$$

Assume $\tilde{h}$ induces the kernel κ on the first d coordinates in the sense that, for Z_ℓ of the form (8),

$$\left[\tilde{h}(KZ_\ell, QZ_\ell)\right]_{j,i} = \begin{cases} \kappa(s^{(j)}, s^{(i+1)}), & i \in \{0, 1, \ldots, n-1\}, \\ \kappa(s^{(j)}, s^{(\texttt{pad})}), & i = n. \end{cases}$$

Then for any Z_ℓ of the form (8),

$$\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z_\ell) = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ y_\ell^{(0)} & \cdots & y_\ell^{(n-1)} & y_\ell^{(n)} \end{bmatrix},$$

where

$$y_\ell^{(i)} \;:=\; \begin{cases} \gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(i+1)}), & i \in \{0, 1, \ldots, n-1\}, \\ \gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(\texttt{pad})}), & i = n. \end{cases}$$

Proof. Write $Z_\ell = \begin{bmatrix} Z_\ell^{(1)} \\ Z_\ell^{(2)} \\ Z_\ell^{(3)} \end{bmatrix}$ where $Z_\ell^{(1)} \in \mathbb{R}^{d\times(n+1)}$ is the first block-row, $Z_\ell^{(2)} \in \mathbb{R}^{d\times(n+1)}$ is the second block-row, and $Z_\ell^{(3)} \in \mathbb{R}^{1\times(n+1)}$ is the last row.

Step 1: KZ_ℓ and QZ_ℓ. By the definition of K, we have

$$KZ_\ell = \begin{bmatrix} Z_\ell^{(1)} \\ \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{1\times(n+1)} \end{bmatrix}.$$

By the definition of Q, we have

$$QZ_\ell = \begin{bmatrix} Z_\ell^{(2)} \\ \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{1\times(n+1)} \end{bmatrix}.$$

Hence, by the assumed kernel-inducing property of $\tilde{h}$ (stated in the lemma),

$$\left[\tilde{h}(KZ_\ell, QZ_\ell)\right]_{j,i} = \begin{cases} \kappa(s^{(j)}, s^{(i+1)}), & i \in \{0,\dots,n-1\}, \\ \kappa(s^{(j)}, s^{(\texttt{pad})}), & i = n. \end{cases}$$

Step 2: VZ_ℓ. By the definition of V, VZ_ℓ is zero in its first $2d$ rows and equals $\gamma\alpha_\ell Z_\ell^{(3)}$ in its last row:

$$VZ_\ell = \begin{bmatrix} \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{d\times(n+1)} \\ \gamma\alpha_\ell Z_\ell^{(3)} \end{bmatrix}.$$

Step 3: masking by M. Multiplying on the right by M zeros out the last (query) column, so

$$VZ_\ell\, M = \begin{bmatrix} \mathbf{0}_{d\times(n+1)} \\ \mathbf{0}_{d\times(n+1)} \\ \gamma\alpha_\ell\,[b_\ell^{(0)},\dots,b_\ell^{(n-1)},0] \end{bmatrix}.$$

Step 4: multiply by affinity matrix. Therefore

$$\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z_\ell) = (VZ_\ell\, M)\,\tilde{h}(KZ_\ell, QZ_\ell)$$

has zero in its first $2d$ rows, and for each $i \in \{0,\dots,n\}$ its last-row entry is

$$\gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)} \left[\tilde{h}(KZ_\ell, QZ_\ell)\right]_{j,i}.$$

Substituting the expression for $H_{j,i}$ yields

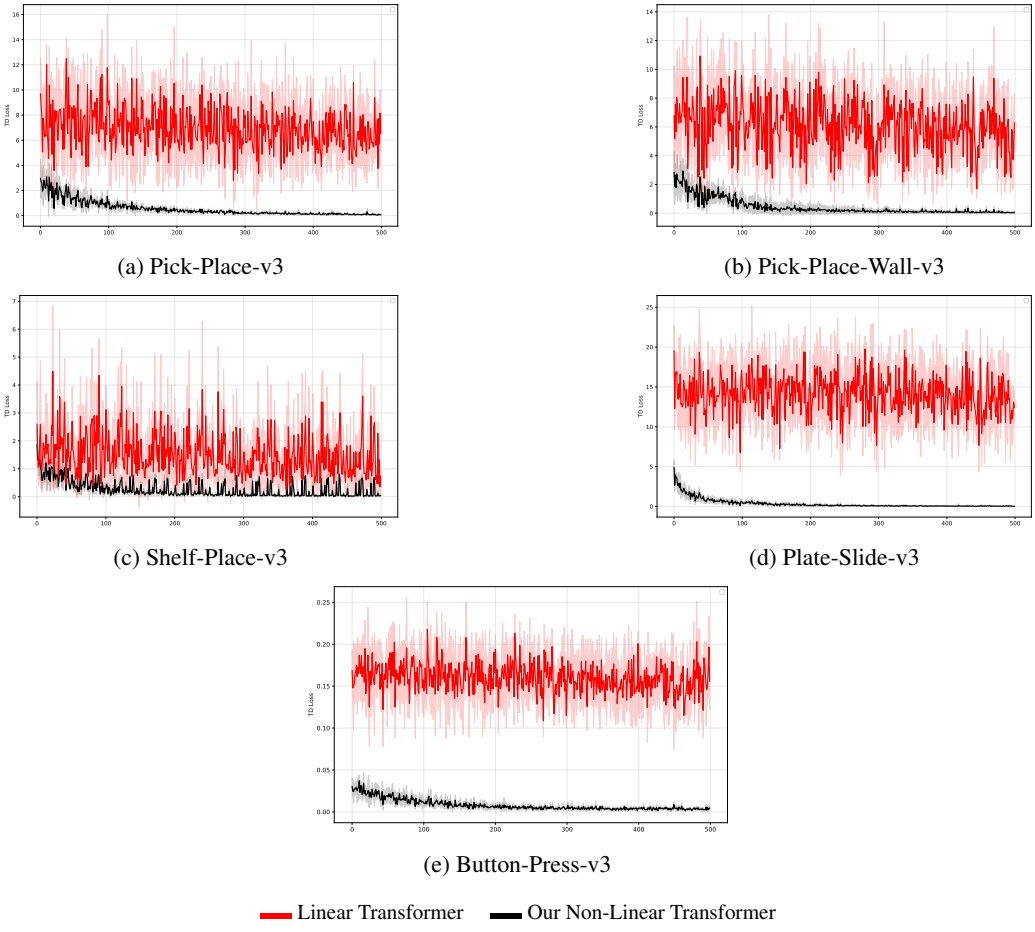
$$\begin{aligned} &\left[\texttt{Attn}^{\tilde{h}}_{K,Q,V}(Z_\ell)\right]_{2d+1,i} \\ &= \begin{cases} \gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(i+1)}), & i \in \{0,\dots,n-1\}, \\ \gamma\alpha_\ell \sum_{j=0}^{n-1} b_\ell^{(j)}\, \kappa(s^{(j)}, s^{(\texttt{pad})}), & i = n, \end{cases} \end{aligned}$$

which is exactly $y_\ell^{(i)}$ as defined in the lemma, completing the proof. □

B Learning Curves for Different Domains

To provide additional evidence supporting our results, we include in this section the learning curves for each domain, averaged over five random seeds. For comparison, we also report the learning curves of the linear transformer baseline from Wang et al. [26].

Specifically, the learning curves report the TD loss values used to optimize the model parameter α, as described in Section 6.1. They illustrate how the adapted learning rate in the induced kernel regression affects the convergence of the TD loss. Figure 4 present the results. We use identical training configurations for both the non-linear and linear transformers. The only difference is that the linear transformer uses the default learning rate of 0.001, as reported by the authors. The failure of the linear transformer to converge on MetaWorld tasks in terms of the TD error is expected, given that it is restricted to a linear function class, which is inadequate for the underlying value functions.



(a) Pick-Place-v3

(b) Pick-Place-Wall-v3

(c) Shelf-Place-v3

(d) Plate-Slide-v3

(e) Button-Press-v3

Figure 4: Learning curves across five MetaWorld domains. Each curve is averaged over five random seeds.

C One Model for All Domains

It is also natural to ask whether a model trained on a single domain can be reused across multiple domains, a question that directly reflects the title of this work. As shown in Appendix E.2, four domains share a common kernel temperature, whereas **Button-Press-v3** requires a distinct setting. This observation suggests that models may generalize effectively across the four domains but fail to transfer to **Button-Press-v3**, a discrepancy whose underlying causes are discussed in Section 5.

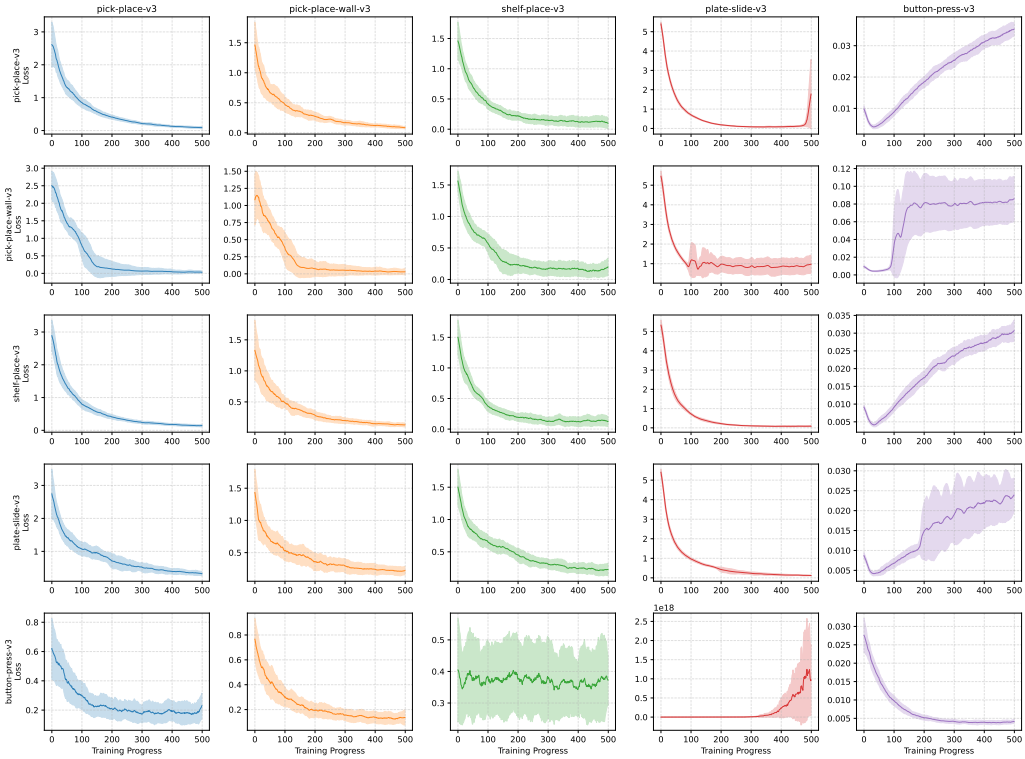


Figure 5: Models trained on a single domain are applied to all domains simultaneously.

To test this hypothesis, we report a cross-domain evaluation matrix averaged over five random seeds, where each row corresponds to a training domain and each column corresponds to an evaluation domain. For example, the first row shows the performance of models trained on **Pick-Place-v3** and evaluated across all five domains using checkpoints saved throughout training. Thus, the subplots along the diagonal correspond to settings where the training and evaluation domains are the same. Importantly, these models are evaluated directly in the target domains without any additional fine-tuning or training. Figure 5 presents the resulting TD loss curves, showing that the TD error continues to decrease across the four domains, while diverging on Button-Press-v3, consistent with our hypothesis.

Specifically, for models trained on **Pick-Place-v3**, **Pick-Place-Wall-v3**, **Shelf-Place-v3**, and **Plate-Slide-v3**, corresponding to the first four rows of the matrix, the TD loss generally continues to decrease when evaluated on the first four domains, corresponding to the first four columns of the matrix. The main exception is that models trained on **Pick-Place-v3** exhibit divergence on **Plate-Slide-v3** toward the end of training. This behavior may arise because **Plate-Slide-v3** exhibits a substantially different value range, and excessive adaptation of the scaling parameter α and the temperature parameter can induce an RKHS that is no longer well aligned with this domain. Nevertheless, the model converges over a large portion of the training trajectory. By contrast, the divergence of models evaluated on **Button-Press-v3** is expected, as discussed above. For models trained on **Button-Press-v3**, corresponding to the last row of the matrix, we observe that the value ranges differ substantially from those of the first four training domains and that the TD loss exhibits significant divergence when transferred to other domains. This further suggests that **Button-Press-v3** is substantially different from the other domains, consistent with our hypothesis.

D Discontinuous State Value Function and TD Error

A key assumption of the model is that the underlying state-value function lies within the induced RKHS, under which the function can be accurately approximated via in-context learning. When this assumption is violated, the model's performance degrades. Under the original MetaWorld reward

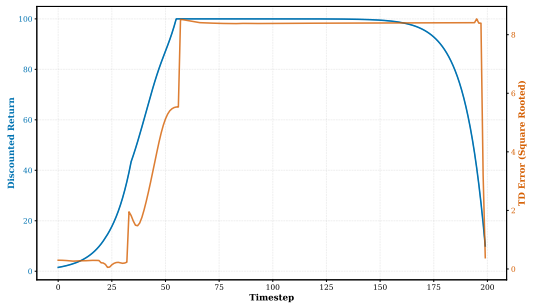


Figure 6: Illustration of the induced state-value function under the original MetaWorld reward settings. The value function exhibits large discontinuities, violating the smoothness assumptions implicit in the RKHS approximation.

settings, the induced state-value function is highly discontinuous, making it difficult for the TD error to converge to zero across all states.

To illustrate this point, we take a fully trained model on Pick-Place-v3 and visualize, along an entire episode, the TD error at each state against the corresponding discounted return. Figure 6 illustrates the result. Note that we report the ***square root*** of the TD loss to place it on a scale comparable to the state value. After this rescaling, the TD error remains on the order of ***10***, which is still substantially smaller than the state-value scale of approximately ***100***. The TD error remains near zero during the initial time steps but increases as the time horizon grows. Overall, **this behavior stems from properties of the environment rather than a fundamental mistake of the model**. Simple modifications such as reward rescaling—which effectively rescales the value-function landscape—can help mitigate this issue. In light of this behavior, we report TD loss values over the initial time steps.

E Details for Experiments

E.1 Environment Setting

To construct the MRPs from the MetaWorld environments, as discussed in Section 6, we inject Gaussian noise at varying levels into the actions generated by the optimal policies. For completeness, we report the corresponding noise standard deviations and the estimated initial state values for each domain. All state values are computed using a discount factor of $\gamma = 0.9$. Table 1 presents the results. Notably, the relationships among the initial state values $V(s_0)$ align closely with the trends observed in Figure 2. For example, in Button-Press-v3, the $V(s_0)$ values are relatively close to one another, which leads to greater overlap among the corresponding curves in Figure 2.

The values span a broader range, reflecting our choice of domains designed to evaluate the model under diverse settings.

E.2 Hyperparameter Settings

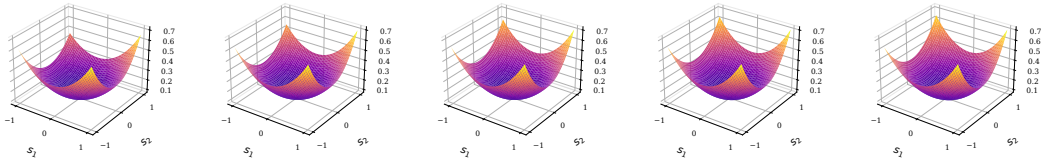
We provide the hyperparameters used across all experiments in this work in table 2. While the domains share a set of common hyperparameters, each domain also includes domain-specific settings. In particular, as discussed in Section 5, the current construction requires different activation temperatures (i.e., kernel parameters) across domains. While our experiments show that a single set of kernel parameters can perform well across multiple domains, we nevertheless report common and domain-specific hyperparameters separately to emphasize that these parameters may still require adjustment. We note that these hyperparameters could be further optimized to yield stronger empirical performance.

Table 1: Gaussian noise levels injected into optimal-policy actions and the corresponding estimated initial state values for each MetaWorld domain.

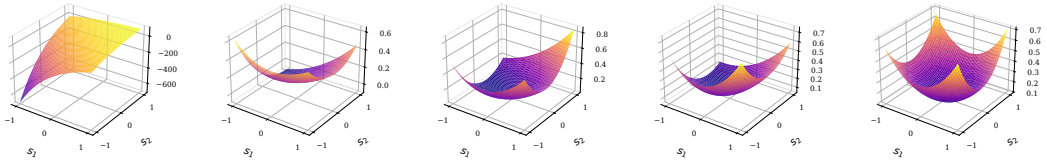
Domain	Noise Std. (σ)	Initial State Value $V(s_0)$
Pick-Place-v3	0.0 (low)	1.52
	0.3 (medium)	1.31
	0.5 (high)	0.79
Pick-Place-Wall-v3	0.0	0.91
	0.2	0.54
	0.4	0.16
Shelf-Place-v3	0.0	2.06
	0.4	1.98
	0.8	1.06
Plate-Slide-v3	0.0	10.33
	0.5	9.89
	0.9	9.12
Button-Press-v3	0.0	3.01
	0.6	2.94
	0.9	2.87

Table 2: Hyperparameters. We report (i) common settings shared across all domains and (ii) domain-specific overrides.

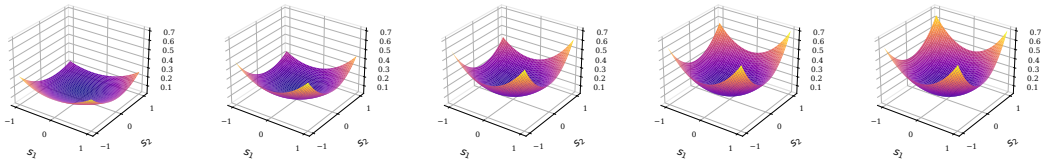
Common (all domains)	
Batch size	32
Optimizer	Adam
Weight decay	1×10^{-6}
Context length	1000
State vector dimensionality	39
Number of transformer layers	30
Activation function	Exponential
Discount factor γ	0.9
Pick-Place-v3	
Activation temperature	10.0
Learning rate	0.01
Pick-Place-Wall-v3	
Activation temperature	10.0
Learning rate	0.01
Shelf-Place-v3	
Activation temperature	10.0
Learning rate	0.01
Plate-Slide-v3	
Activation temperature	10.0
Learning rate	0.01
Button-Press-v3	
Activation temperature	2.5
Learning rate	0.0001



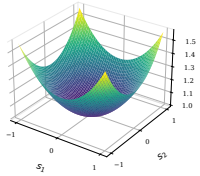
(a) Model-approximated state value functions across random seeds.



(b) Model-approximated state value functions as the number of context transitions varies. From left to right, the number of context transitions is 2, 4, 8, 16, and 32.



(c) Model-approximated state value functions as the number of transformer layers varies. From left to right, the number of context transitions is 2, 4, 8, 16, and 32.



(d) Ground-truth state value function.

Figure 7: Comparison between the model-approximated and ground-truth state value functions.

F Synthetic Environment

F.1 Additional Illustrations

We provide additional illustrations to examine the empirical results on the synthetic environment from multiple perspectives. Figure 7a shows the model-approximated state value functions under different random seeds. For each seed, the context is sampled independently. Across these different contexts, the transformer consistently recovers the shape of the state value function with high accuracy.

We further conduct ablation studies on context length and model depth. Figure 7b varies the number of transitions provided to the transformer while fixing the number of layers at 32, whereas Figure 7c varies the number of transformer layers while fixing the number of context transitions at 32. In both studies, we sweep over $2, 4, 8, 16, 32$. These results show how the quality of the value-function approximation changes as the amount of context and the number of forward-pass TD update steps increase, consistent with our expectation that sufficient context and model depth are necessary for the model to achieve satisfactory performance.

F.2 Environment Details

We provide details on the synthetic MRP used in our experiments. The purpose of this environment is to construct a controlled setting in which the ground-truth state value function lies in an RKHS and can therefore be theoretically approximated via kernel regression. If the transformer can recover the

underlying state value function from environment transitions provided as context, this empirically supports our claim that the transformer implements kernel TD regression in its forward pass.

State space and transition dynamics. We define a Markov Reward Process (MRP) as a tuple

$$\mathcal{M} = (\mathcal{S}, P, r, \gamma),$$

where the state space is the two-dimensional compact domain

$$\mathcal{S} = [-1, 1]^2 \subset \mathbb{R}^2.$$

Given a current state $s_t \in \mathcal{S}$, the next state is generated according to the transition kernel $P(\cdot \mid s_t)$ induced by the linear Gaussian dynamics

$$s_{t+1} = \rho s_t + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I_2), \tag{10}$$

where $\rho \in (0, 1)$. Equivalently,

$$P(\cdot \mid s) = \mathcal{N}\left(\rho s, \sigma^2 I_2\right).$$

The contraction parameter $\rho < 1$ encourages trajectories to remain concentrated around the origin, while the Gaussian noise introduces stochasticity into the state evolution. In our experiments, we set $\rho = 0.5$ and $\sigma = 0.2$, which produces smooth stochastic trajectories from any sampled initial states. We use a discount factor of $\gamma = 0.9$, consistent with the experimental setting used in MetaWorld.

Under these linear Gaussian dynamics, the Markov chain admits a stationary distribution

$$\mathcal{N}\left(0, \frac{\sigma^2}{1 - \rho^2} I_2\right).$$

The two-dimensional state-space construction makes it possible to visualize both the ground-truth value function and the transformer-approximated value function as three-dimensional surface plots.

Kernel and landmark construction. We use the same exponential kernel as the one induced by the nonlinear attention activation:

$$\kappa(x, y) = \exp\left(\frac{x^\top y}{\delta}\right),$$

where $\delta > 0$ denotes the kernel temperature. In our experiments, we set $\delta = 1$.

To construct the ground-truth value function, we place m centroid points uniformly on the unit circle:

$$c_j = \left(\cos\left(\frac{2\pi j}{m}\right), \sin\left(\frac{2\pi j}{m}\right)\right), \qquad j = 0, \ldots, m-1.$$

In our experiments, we set $m = 8$. This symmetric configuration induces a smooth value landscape, making it well suited for visualization. Figure 8 visualizes the centroid points, which are evenly distributed along the unit circle.

Ground-truth value function. We define the ground-truth state value function as a uniform mixture of kernel functions centered at the landmark points. Specifically, for every $s \in [-1, 1]^2$,

$$V^\star(s) = \frac{1}{m} \sum_{j=0}^{m-1} \kappa(c_j, s) = \frac{1}{m} \sum_{j=0}^{m-1} \exp\left(\frac{c_j^\top s}{\delta}\right).$$

Intuitively, $V^\star(s)$ measures the aggregate kernel similarity between the state s and the centroid points. Since the centroids are uniformly distributed along the unit circle, states with larger norms tend to align more strongly with a subset of the centroids. Consequently, the value function increases as the state moves away from the origin, with the largest values attained near the corners of the state space. Figure 8 visualizes this effect.

By construction, $V^\star \in \mathcal{H}_\kappa$, since it is a finite linear combination of kernel evaluations.

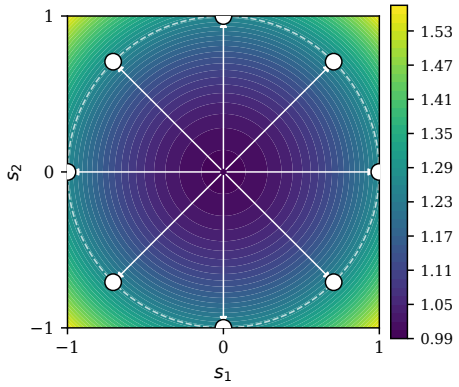


Figure 8: Selected centroid points and the induced state value function.

Bellman-consistent reward. We derive the reward function from the desired value function so that the Bellman equation is satisfied exactly. For a single kernel term, using the Gaussian transition dynamics gives

$$\begin{aligned}\mathbb{E}\left[\kappa(c_j, s_{t+1}) \mid s_t = s\right] &= \mathbb{E}\left[\exp\left(\frac{c_j^\top(\rho s + \epsilon_t)}{\delta}\right)\right] \\ &= \exp\left(\frac{(\rho c_j)^\top s}{\delta}\right)\exp\left(\frac{\|c_j\|^2\sigma^2}{2\delta^2}\right).\end{aligned}$$

Since the centroids lie on the unit circle, $\|c_j\| = 1$, and therefore the Gaussian correction term is the same for all landmarks. Denote

$$\eta = \exp\left(\frac{\sigma^2}{2\delta^2}\right).$$

Then

$$\mathbb{E}\left[V^\star(s_{t+1}) \mid s_t = s\right] = \frac{\eta}{m}\sum_{j=0}^{m-1}\kappa(\rho c_j, s).$$

We define the reward function by rearranging the Bellman equation $V^\star(s) = R(s) + \gamma\mathbb{E}[V^\star(s_{t+1}) \mid s_t = s]$:

$$R(s) = \frac{1}{m}\sum_{j=0}^{m-1}\left[\kappa(c_j, s) - \gamma\eta\kappa(\rho c_j, s)\right]. \tag{11}$$

Substituting into the Bellman equation gives

$$\begin{aligned}R(s) + \gamma\mathbb{E}\left[V^\star(s_{t+1}) \mid s_t = s\right] &= \frac{1}{m}\sum_{j=0}^{m-1}\left[\kappa(c_j, s) - \gamma\eta\kappa(\rho c_j, s)\right] + \frac{\gamma\eta}{m}\sum_{j=0}^{m-1}\kappa(\rho c_j, s) \\ &= \frac{1}{m}\sum_{j=0}^{m-1}\kappa(c_j, s) = V^\star(s).\end{aligned}$$

Thus, the constructed reward function can be exactly calculated for each state and is Bellman-consistent with $V^\star$. Moreover, $R \in \mathcal{H}_\kappa$, since it is also a finite linear combination of kernel functions.

Transformer Input. For each prompt, we uniformly sample a set of context states $(s_0, s_1, \ldots, s_{n-1})$ from the synthetic state space. These states are then used to generate the corresponding next states $(s'_0, s'_1, \ldots, s'_{n-1})$ and rewards $(r_0, r_1, \ldots, r_{n-1})$ according to the transition dynamics in Equation 10 and the reward function in Equation 11. An arbitrary query state is then appended to the context to form Z_0, which is provided as input to the transformer for value prediction. The transformer performs kernel TD updates through its forward pass, using the context states as kernel centers.

Importantly, the input provided to the transformer consists only of individual transition samples. The context states are uniformly distributed across the state space and are not chosen to match the centroid configuration used to define $V^\star$. Thus, recovering the underlying state value function requires the transformer to correctly implement the TD update procedure from the sampled transitions.

In the experiment shown in Figure 3, we uniformly sample 32 context states from the state space together with their corresponding next states and rewards. We then fix the context and vary the query state over a grid in $[-1, 1]^2$, allowing us to visualize the transformer-predicted value surface. We use 30 transformer layers, consistent with the MetaWorld experiments. As shown in the results, the transformer accurately recovers the underlying state value function, up to the constant bias discussed above.

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 explicitly included the contribution and scope.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We explicitly included in section 5 to discuss about limitations.

   Guidelines:

   - The answer [N/A] 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 provide formal settings and full derivations in section 4 and appendix A.

Guidelines:

- The answer [N/A] 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 experiment details in appendix E.2 and appendix F.

   Guidelines:

   - The answer [N/A] 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 provide the code as supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: We include experiment details in appendix E.2 and appendix F.

   Guidelines:

   - The answer [N/A] 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 include experiments with random seeds and sufficient stochasticity.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: The work requires only minimal computational resources for training, and we believe it can be reproduced with standard computing capacity.

   Guidelines:

   - The answer [N/A] 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 comply with the code of ethics.

   Guidelines:

   - The answer [N/A] 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: [No]

    Justification: The work is purely algorithmic, and to the best of our knowledge, it does not raise any direct societal impact concerns.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: The work is purely algorithmic, and to the best of our knowledge, it does not raise any direct safety concerns.

    Guidelines:

    - The answer [N/A] 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 wrote the code by our own.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: The code is well structured and includes a README file.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The work is purely algorithmic, and to the best of our knowledge, it does not involve any human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The work is purely algorithmic, and to the best of our knowledge, it does not involve any human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for editing and sentence polishing. The methodology, scientific rigor, and originality of the research are entirely the authors' own.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.