TASK DIVERSITY SHORTENS THE ICL PLATEAU

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) describes a language model's ability to generate outputs based on a set of input demonstrations and a subsequent query. To understand this remarkable capability, researchers have studied simplified, stylized models. These studies have consistently observed long loss plateaus, during which models exhibit minimal improvement, followed by a sudden, rapid surge of learning. In this work, we reveal that training on multiple diverse ICL tasks simultaneously *shortens* the loss plateaus, making each task easier to learn. This finding is surprising as it contradicts the natural intuition that the combined complexity of multiple ICL tasks would lengthen the learning process, not shorten it. Our result suggests that the recent success in large-scale training of language models may be attributed not only to the richness of the data at scale but also to the easier optimization (training) induced by the diversity of natural language training data.

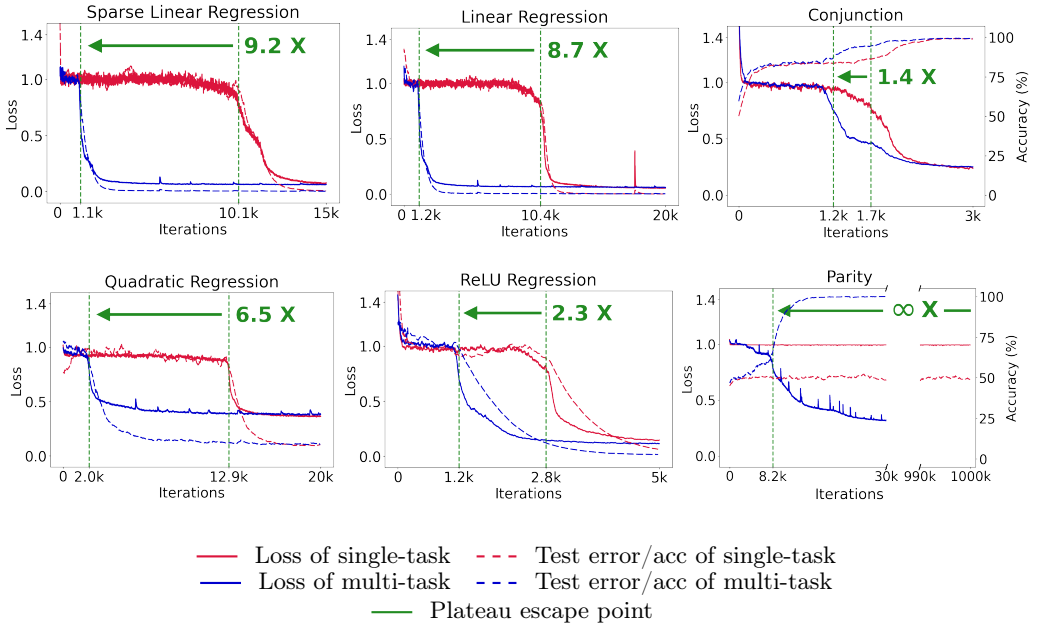


Figure 1: We train a transformer from scratch on in-context learning tasks. Single-task ICL: Training loss (——) and test error/accuracy (- - -) when each task is trained individually. The Parity task cannot be learned even after 1000k training steps. Multi-task ICL: Training loss (——) and test error/accuracy (- - -) when all six tasks are trained simultaneously. Green lines mark the plateau escape points. Surprisingly, multi-task ICL training significantly shortens these plateaus, making training easier.

1 INTRODUCTION

In-context learning (ICL), first reported by Brown et al. (2020) with GPT-3, describes a language model's ability to generate outputs based on a set of input demonstrations and a subsequent query. In ICL, the model discerns the task implied by the context of these demonstrations without explicit descriptions, indicating that the model may internally implement an *algorithm* or engage in a *reasoning process*. To understand this remarkable capability that emerges in language models trained on complex real-world language data, researchers such as Garg et al. (2022) have studied simplified, stylized models. In these studies, transformers are trained from scratch to learn simple functions in context, such as linear regression. We thoroughly review this prior work in Section 1.1.

An intriguing phenomenon observed in these works is the *long loss plateau* in training for ICL. Throughout these plateaus, models display minimal performance improvement, followed by a sudden, rapid surge of learning—deviating from the typical smooth reduction in training loss. In this work, we reveal that **training on multiple ICL tasks simultaneously shortens the loss plateaus** as illustrated in Figure 1. This finding is surprising as it contradicts the natural intuition that the combined complexity of multiple ICL tasks would lengthen the learning process, not shorten it.

In the language of multi-task learning, our findings present an instance where multi-task learning is *easier* than single-task learning in the sense of training dynamics. While previous research has primarily focused on the *statistical* benefits of task diversity in multi-task learning, our findings reveal *optimization* benefits. This insight suggests that the recent success in large-scale training of language models may be attributed not only to the richness of the data at scale but also to the easier optimization (training) induced by the diversity of natural language training data.

Organization. Section 2 describes the experimental setup for training for multiple ICL tasks. Section 3 articulates our main claim that task diversity shortens the ICL plateau and presents experimental evidence with transformers and state-space models trained on synthetic and natural language ICL tasks. Section 4 investigates the underlying reasons for this phenomenon. Section 4.1 characterizes the model's behavior during the plateau, which we refer to as the no-context learning regime. Section 4.2 shows that there is a common structure shared across the ICL tasks and that task diversity accelerates the learning of this shared structure. Section 5 presents further experimental details. Section 6 concludes the paper.

1.1 RELATED WORKS

In-context learning. In-context learning (ICL) abilities of pretrained Large Language Models have gained significant attention since first investigated by GPT-3 (Brown et al., 2020). Following this, a large body of empirical studies have explored ICL in Large Language Models (Min et al., 2022a;b; Liu et al., 2021; Nie et al., 2022; Rubin et al., 2022; Wei et al., 2023). Given the complexity of real-world data, researchers have explored ICL in more stylized and simplified setups. Garg et al. (2022) formalized an approach to studying transformer's performance to learn simple function class in context. Building on this work, several works have investigated the ability of models to learn stylized function classes in various settings, including boolean functions (Bhattamishra et al., 2024), regular language (Akyürek et al., 2024), as well as task mixtures (Tripuraneni et al., 2023), and the ability of Mamba (Park et al., 2024; Grazzi et al., 2024; Li et al., 2024). For more comprehensive overview, please refer to the survey by Dong et al. (2024).

While these studies have scrutinized ICL abilities across various setups, most have not delved into the in-context algorithms that are implemented by models. Xie et al. (2022) suggested that the ICL process can be interpreted as Bayesian inference. A number of works have argued, both theoretically and empirically, that transformers implement gradient descent to learn linear regression in-context (Akyürek et al., 2023; von Oswald et al., 2023; Mahankali et al., 2024; Zhang et al., 2024; Ahn et al., 2023). Beyond these works, many have sought to uncover the internal ICL procedure of transformers, in more complex algorithms (Fu et al., 2023; Giannou et al., 2024; Cheng et al., 2024; von Oswald et al., 2024; Lin & Lee, 2024); when handling more intricate tasks (Wang et al., 2024; Guo et al., 2024; Bai et al., 2023; Lin et al., 2024; Wang et al., 2024).

Abrupt phase transition in in-context learning. Many studies (Srivastava et al., 2023; Wei et al., 2022; He et al., 2024; Chan et al., 2022; Raventós et al., 2023; Raventos et al., 2023) have shown

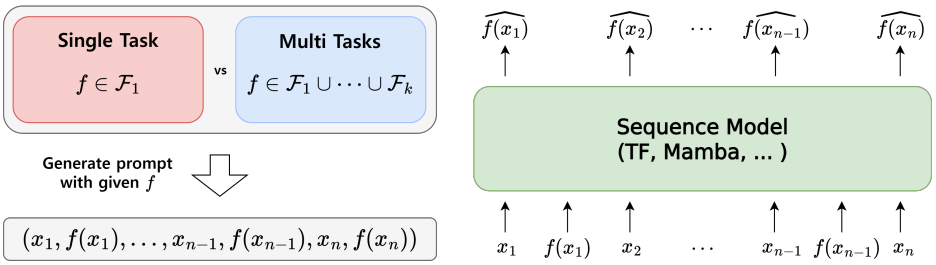


Figure 2: To generate a training sequence, we sample the function $f \in \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_k$, where $\mathcal{F}_1, \ldots, \mathcal{F}_k$ are different in-context function classes, sample $x_1, \ldots, x_n$ IID, and form $(x_1, f(x_1), \cdots, x_{n-1}, f(x_{n-1}), x_n, f(x_n))$. We refer to the case $k = 1$ as single-task ICL and $k > 1$ as multi-task ICL. The sequence model is trained with autoregressive next-token prediction, i.e., the model predicts $f(x_i)$ conditioned on $(x_1, f(x_1), x_{i-1}, f(x_{i-1}), \cdots, x_i)$ for $i = 1, \ldots, n$.

that a model's ability to perform ICL emerges abruptly, with respect to dataset and model size. While these studies do not focus on the training process, abrupt performance gains during training have also been reported in various works. This transition is typically associated with escaping *loss plateaus*. During these plateaus, no performance gains are observed, but once the plateau is escaped, the model begins to learn in-context abruptly. This phenomenon has been observed in various setups (Garg et al., 2022; Bhattamishra et al., 2024; Park et al., 2024; Li et al., 2024; Kirsch et al., 2024), although these works did not primarily focus on it.

Beyond these, several studies have explored loss plateaus themselves. Fu et al. (2024) suggested that models learn the features of dataset during loss plateaus. Chen et al. (2024) theoretically demonstrated that a plateau occurs during training when a one-layer transformer is trained by linear regression ICL task. Olsson et al. (2022) proposed that the plateau escape and the formation of 'induction head' simultaneously happen. Further work by Reddy (2024); Singh et al. (2024); Song et al. (2024), focused on two-layer transformers, explicitly defining the induction head with transformer parameters and characterizing the internal mechanisms behind its formation. In contrast, Section 4.1 offers a new perspective by providing the explicit form of the model's output during plateaus. Furthermore, in Section 4.2.3, we provide a discrepancy between our finding and induction head.

Task diversity. Multi-task training (Caruana, 1997; Baxter, 2000) is a widely used approach for model pretraining. In particular, researchers have identified that *task diversity* is crucial for pretrained models to outperform in downstream tasks, alongside a large body of research across various domains: supervised learning (Tripuraneni et al., 2020; 2021; Du et al., 2021; Maurer et al., 2016; Crawshaw, 2020; Ruder, 2017), reinforcement learning (Zhang et al., 2023a; Jin et al., 2020; Hu et al., 2021; Yang et al., 2021; Collins et al., 2021; Lu et al., 2022; Cheng et al., 2022), and natural language processing (Zhang et al., 2023a; Zhao et al., 2023; Zhang et al., 2023b; Hu et al., 2020; Song et al., 2020; Zhou et al., 2019; Gunasekar et al., 2023; Sharma et al., 2023). These works mostly focused on statistical benefits, whereas our finding emphasize on the optimization benefits.

2 Experimental setup

Our experimental setup follows Garg et al. (2022) and Bhattamishra et al. (2024). Consider a function class $\mathcal{F}$ with domain $\mathcal{X}$. A sequence model M_θ (transformer or state-space model) is trained to identify $f \in \mathcal{F}$ in context and make a prediction on the subsequent query. The training data consists of sequences of the form $P = (x_1, f(x_1), x_2, f(x_2), \ldots, x_{n-1}, f(x_{n-1}), x_n, f(x_n))$, where f is sampled from a distribution $\mathcal{D}_\mathcal{F}$ and $x_1, \ldots, x_n$ are independently sampled from a distribution $\mathcal{D}_\mathcal{X}$. We train M_θ for the next token-prediction task: The loss over the sequence P is given by $\frac{1}{n}\sum_{i=1}^{n} \ell(M_\theta(P_i), f(x_i))$, where $\ell(\cdot, \cdot)$ is an appropriate loss function and $P_i := (x_1, f(x_1), x_2, f(x_2), \ldots, x_{i-1}, f(x_{i-1}), x_i)$ is the i-th prefix for $i = 1, \ldots, n$. In our experiments, $n = 120$ is a predetermined number shared across all tasks. This procedure is illustrated on Figure 2.

ICL tasks. We consider *continuous ICL tasks* and *boolean ICL tasks*. For each ICL task, an $f \in \mathcal{F}$ is chosen, where $\mathcal{F}$ is a function class. For continuous ICL tasks, $\mathcal{F}$ consists of $f\colon \mathbb{R}^d \to \mathbb{R}$ for $d \in \mathbb{N}$ and the probability distribution on the domain is assumed to be $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_d)$. For boolean ICL tasks, $\mathcal{F}$ consists of $f\colon \{\pm 1\}^d \to \{\pm 1\}$ for $d \in \mathbb{N}$ and the probability distribution on the domain is assumed to be $\mathcal{D}_{\mathcal{X}} = \mathrm{Unif}\left(\{\pm 1\}^d\right)$. We consider 10 different ICL tasks: Linear Regression, Quadratic Regression, Sparse Linear Regression, ReLU Regression, Decision Tree, Sparse Parity(2), Sparse Parity(3), Parity, Conjunction, and Disjunction. The function class $\mathcal{F}$ defining each of these 10 ICL tasks is precisely stated in Section 5.

Multi-task ICL training. Most prior work on ICL, such as Garg et al. (2022); von Oswald et al. (2023); Bhattamishra et al. (2024); Park et al. (2024), considers ICL training with a single function class $\mathcal{F}$. In this work, we train models to learn functions from the union of function classes $\bigcup_{m=1}^{k} \mathcal{F}_m$ in context, where $\mathcal{F}_1, \ldots, \mathcal{F}_k$ are distinct ICL task among the 10 that we list in Section 5. For instance, if the model is trained on the union of linear and quadratic regression tasks, then $\mathcal{F}_1 \cup \mathcal{F}_2 = \{f \mid f(x) = w^\intercal x\} \cup \{f \mid f(x) = x^\intercal W x\}$. For each $m = 1, \ldots, k$, we sample $f \sim \mathcal{D}_{\mathcal{F}_m}$ and $x_1, \ldots, x_n \overset{\text{IID}}{\sim} \mathcal{D}_{\mathcal{X}_m}$ and form the sequence $(x_1, f(x_1), x_2, f(x_2), \ldots, x_n, f(x_n))$. This sampling process is repeated B times for each $m = 1, \ldots, k$, making kB the total batch size. To balance the loss scales across different tasks, we normalize each task's loss with constant factors $c_1, \ldots, c_k$ (further discussed in Section 4.1) so that the training loss stabilizes around 1 during the plateaus. Appendix A provides further experimental details. In expectation, we minimize the loss

$$L(\theta) = \sum_{m=1}^{k} c_m \mathop{\mathbb{E}}_{\substack{f \sim \mathcal{D}_{\mathcal{F}_m} \\ x_1, \ldots, x_n \overset{\text{IID}}{\sim} \mathcal{D}_{\mathcal{X}_m}}} \left[\frac{1}{n} \sum_{i=1}^{n} \ell\big(M_\theta(P_i), f(x_i)\big) \right].$$

Test loss. To evaluate the model's performance on ICL tasks, we measure the error of the model's prediction on the last (n-th) output of the prompt. For continuous ICL tasks, the test error is $(M_\theta(P_n) - f(x_n))^2$. For boolean ICL tasks, the test accuracy is measured by $\mathbf{1}[\mathrm{sign}(M_\theta(P_n)) = f(x_n)]$, where $\mathbf{1}$ is the indicator function.

3 Task diversity shortens ICL plateaus

Long loss plateaus have been commonly reported in the various setups for training sequence models from scratch to perform ICL, including simple in-context functions (Garg et al., 2022; Chen et al., 2024; Li et al., 2024; Bhattamishra et al., 2024; Park et al., 2024), image datasets (Fu et al., 2024; Kirsch et al., 2024; Singh et al., 2024; Reddy, 2024), and language datasets (Akyürek et al., 2024; Olsson et al., 2022). In this section, we present the following claim:

Claim: Task diversity shortens the ICL plateau, making each task easier to learn.

Here, task diversity refers to learning multi-task ICL on a mixture of distinct function classes. The claim that multi-task ICL is easier to learn than single-task ICL is surprising as it contradicts the natural intuition that the combined complexity of multiple ICL tasks would lengthen the learning process, not shorten it.

We provide comprehensive experimental evidence to support this claim. Table 1 summarizes our experimental results on transformers. Figure 3 shows the results of state space models, specifically Mamba (Gu & Dao, 2024) and Hyena (Poli et al., 2023). Across the hundreds of task combinations in these setups, we consistently observed that task diversity enables training to escape plateaus more quickly. Figure 7 shows results in natural language ICL tasks reinforcing our claim. Refer to Appendix A for experimental details and Appendix B for additional tables.

However, not all ICL tasks mutually reduce the duration of plateaus. For instance, we consider Regbench task (Akyürek et al., 2024), which is generated by a random automata. Our experiments in Appendix C.2 show that combining Regbench with the ICL tasks we consider does not reduce the plateau of Regbench, but does reduce the plateau of the other ICL tasks.

Therefore, the claim should be understood as a description of a general tendency rather than a universal law. Nevertheless, our finding is broadly and robustly observed, as borne out by our extensive experiments.

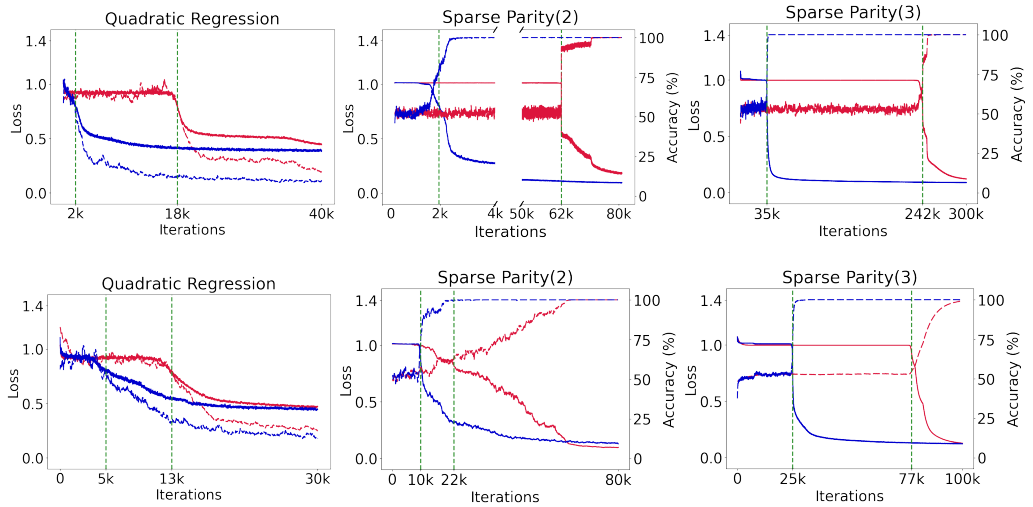


Figure 3: **Mamba (First row)** and **Hyena (Second row)**. Red lines and Blue lines respectively represent the loss dynamics of single-task training and multi-task training. (Left): Quadratic Regression+Linear Regression compared to Quadratic Regression. (Middle): Sparse Parity(2)+Linear Regression compared to Sparse Parity(2). (Right): Sparse Parity(3)+Conjunction compared to Sparse Parity(3).

Number of tasks	Boolean Tasks				Continuous Tasks				
	Sparse Parity(2)	Sparse Parity(3)	Conjunction	Disjunction	Linear Regression	Quadratic Regression	ReLU Regression	Sparse Linear Regression	Decision Tree
1	> 1000k	> 1000k	2.4k (2.7k)	2.5k (2.7k)	9.0k (12.1k)	13.1k (14.2k)	3.0k (3.7k)	16.5k (18.4k)	4.3k (16.0k)
2	12.2k (17.2k)	22.4k (26.7k)	1.6k (1.9k)	1.3k (1.6k)	2.7k (3.4k)	5.5k (7.5k)	2.2k (2.8k)	3.1k (3.8k)	5.9k (19.3k)
4	2.1k (2.4k)	2.9k (3.9k)	1.5k (3.0k)	1.2k (1.5k)	1.4k (2.1k)	2.1k (4.5k)	1.8k (2.5k)	1.4k (2.1k)	2.5k (18.0k)
8	1.5k (2.0k)	2.1k (2.4k)	0.9k (1.5k)	1.0k (1.5k)	0.9k (2.0k)	1.7k (5.4k)	0.9k (2.0k)	0.9k (1.9k)	2.0k (20.0k)
9	1.6k (1.8k)	2.2k (2.3k)	0.9k (1.5k)	0.9k (1.4k)	0.8k (1.8k)	1.6k (6.6k)	0.9k (1.9k)	0.8k (1.8k)	1.8k (22.0k)

Table 1: **Task diversity shortens the ICL plateau**. We train a transformer with various combinations of 9 different tasks ($d = 10$). For each run, we report two metrics: the time to escape the plateau and the time to complete the learning of the task (written in parentheses). The rows correspond to the number of tasks trained together, and each entry in the table corresponds to the average time across the training runs that include the given task. For example, the entry at (Number of tasks = 4, Conjunction) shows the average of $\binom{8}{3}$ results. We find that multi-task training shortens the ICL plateau. Precise details are provided in Appendix A.

Model of task complexity. We quickly describe our mental model of the aggregate complexity of multi-task ICL training and the speedup due to task diversity. Let the 'complexity' of an ICL task be the time it takes to escape from the plateau. The complexity of a single-task ICL is the complexity observed when training with just the single task. The complexity of a multi-task ICL is the sum of the complexities of the constituent tasks, but the task diversity reduces it. This reduction makes the aggregate complexity of the multi-task ICL less than the complexity of the individual single-task ICLs. Section 4.2 discusses when and why task diversity could reduce individual complexities. Figure 4 illustrates this notion.

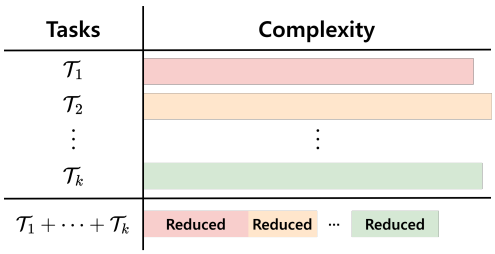


Figure 4: Complexity model of multi-task training

Escaping plateau ≈ training completion. In the ICL setups we consider, we observe that (i) there is only one plateau, and (ii) learning is very rapid once this plateau is escaped from. This implies that the time at which training is completed, defined as the moment when the model reaches near-perfect training accuracy, is very close to the time of escaping from the (first) plateau. The results of Table 1 confirm this.

It should be noted that previous studies have demonstrated that multi-stage learning does occur in supervised learning, both theoretically (Ghosh et al., 2022; Bietti et al., 2022; Jin et al., 2023; Wang & Ma, 2023; Berthier et al., 2024) and empirically (Nakkiran et al., 2019; Refinetti et al., 2023; Rubruck et al., 2024). Therefore, we expect that ICL tasks involving complex hierarchical structures may exhibit multiple plateaus.

4 Why is task diversity helpful?

In Section 3, we demonstrated that task diversity shortens plateaus but did not explore *why* this effect occurs. In this section, we provide partial answers and hypotheses toward understanding this phenomenon.

4.1 Plateau is task-wise no-context learning

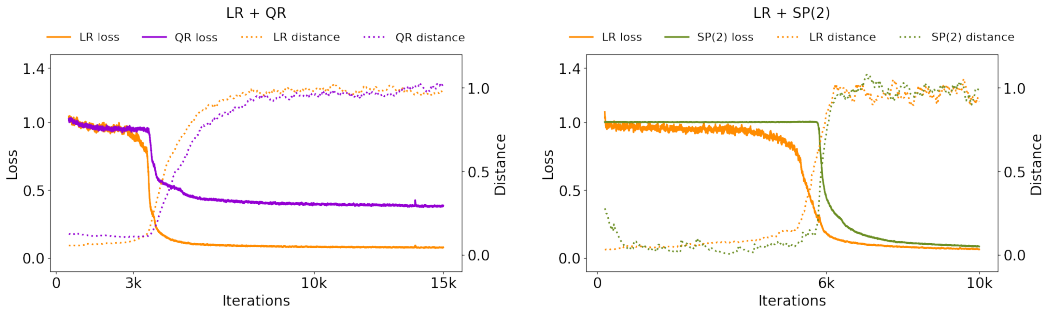


Figure 5: During the plateau, the model output very closely matches the task-wise optimal no-context function. Solid lines (——) denote the raining loss while the dotted lines (- - -) denote the squared distance between model output and task-wise optimal no-context function. **(Left)**: Linear Regression($\mu = -0.5$) + Quadratic Regression($\mu = 0.5$). **(Right)**: Linear Regression($\mu = 1$) + Sparse Parity(2).

At first glance, plateaus might appear to be a failure mode where no meaningful learning occurs. However, in the Sparse Parity(2) task, for example, both test and train accuracies hover around 0.55 during plateaus, as illustrated in Figure 3. If the model learned nothing and were making random predictions, the accuracy should be 0.5. This deviation implies the model is learning something.

For continuous ICL tasks, define the optimal *no-context function* as

$$g^\star_{\mathcal{F}} := \arg\min_g \mathop{\mathbb{E}}_{f\sim\mathcal{D}_{\mathcal{F}}, x\sim\mathcal{D}_{\mathcal{X}}} \left[(g(x) - f(x))^2\right],$$

i.e., $g^\star_{\mathcal{F}}$ is the context-independent function that minimizes the test error. For boolean ICL tasks, $g^\star_{\mathcal{F}}$ is analogously defined with the argmax of the test accuracy. In many cases, the optimal no-context function has a simple closed-form expression. For instance, if $\mathcal{F} = \{f \mid f(x) = w^\intercal x\}$ and $\mathcal{D}_{\mathcal{F}}$ is given by $w \sim \mathcal{N}(\mu, I_d)$, then $g^\star_{\mathcal{F}}(x) = \mu^\intercal x$. As another example, if $\mathcal{F} = \{f \mid f(x) = x^\intercal W x\}$ and $\mathcal{D}_{\mathcal{F}}$ is given by $W \sim \mathcal{N}(U, I_{d\times d})$, then $g^\star_{\mathcal{F}}(x) = x^\intercal U x$. The ICL plateau corresponds to *task-wise no-context learning*, which we describe in the following. When we train a model M_θ for ICL with function classes $\mathcal{F}_1, \ldots, \mathcal{F}_k$, the model's output during its plateau corresponds to

$$M_\theta(P_n) = g^\star_{\mathcal{F}_m}(x_n), \quad P_n \text{ is sampled from } \mathcal{F}_m.$$

In other words, the model identifies the function class $\mathcal{F}_m \in \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$ and then applies the optimal no-context function corresponding to $\mathcal{F}_m$. The in-context demonstrations $(x_1, f(x_1), \ldots, x_{n-1}, f(x_{n-1}))$ are used to determine the function class $\mathcal{F}_m \in \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$ but

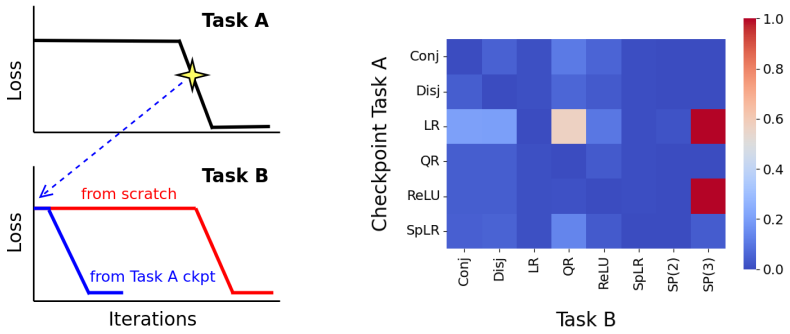


Figure 6: **(Left) Illustration**. We pre-train on Task A and extract a checkpoint ✧ as training escapes the plateau. We then train on Task B starting from the checkpoint. **(Right) Plateau escape time comparison**. Each cell represents the ratio of plateau escape time, with lower ratios (blue color) indicating that the model pre-trained on Task A significantly aids the learning of Task B.

not to determine the specific $f \in \mathcal{F}_m$. This claim can be verified by measuring the error between $M_\theta(P_n)$ and $g^\star_{\mathcal{F}}$, which we plot in Figure 5. Revisiting the Sparse Parity(2) task, the accuracy during plateau is indeed attributed to $0.55 = \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}, x\sim\mathcal{D}_{\mathcal{X}}}[\mathbf{1}(g^\star_{\mathcal{F}}(x) = f(x))]$. Appendix F provides further details on no-context learning regime.

In Section 2, we used the scaling factors $c_1, \ldots, c_k$ to normalize each task's empirical loss. Specifically, we set $c_m = 1/\mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}, x\sim\mathcal{D}_{\mathcal{X}}}\left[\ell(g^\star_{\mathcal{F}_m}(x), f(x))\right]$. Our findings on no-context learning imply that the normalized loss will have a plateau of height 1.

4.2 Common structure across ICL tasks

Consider a multi-task ICL setup, where a model is trained on a set of tasks $\bigcup_{m=1}^k \mathcal{T}_m$. Suppose there exists a "common structure" $\mathcal{C}$ shared across tasks. Denoting the remaining part of each task as $\mathcal{I}_m$, we can decompose each task as $\mathcal{T}_m = \mathcal{C} + \mathcal{I}_m$ for $m = 1, \ldots, k$. Thus, multi-task ICL training is decomposed into two sub-problems: [learning $\mathcal{C}$] and [learning $\mathcal{I}_1, \ldots, \mathcal{I}_k$]. We argue that the main claim of Section 3 can be explained by the following hypotheses:

(1) There exists a common structure shared across the multiple ICL tasks.

(2) The ICL plateau arises from the difficulty of learning this common structure.

(3) Training multiple tasks jointly with a shared structure makes it easier to learn that structure.

In the following, we present evidence supporting these hypotheses. Section 4.2.1 demonstrates (1) and (2). Section 4.2.2 elucidates (3) with a toy experiment on feature learning.

4.2.1 Checkpoint experiment

Consider the following checkpoint experiment. For each single-task training with task A, we save the model as it escapes the plateau as illustrated in Figure 6 (left). Using this checkpoint model as the initialization, we train it on task B.

The findings, summarized in Figure 6, indicate that the checkpoint model transferred from task A quickly learns task B with a shortened plateau. This implies that (1) task A and task B share a common structure and that (2) the plateau arises from the difficulty of learning the common structure. We conduct an analogous experiment on natural language ICL tasks and observe similar results, as shown in Figure 8 of the appendix. Further details are presented in Appendix D.

4.2.2 Generality of our hypothesis

Consider the following intuition. When $\bigcup_{m=1}^k \mathcal{T}_m$ are trained concurrently, the model receives multiple "views" of the common structure $\mathcal{C}$ through the different compositions $\mathcal{C} + \mathcal{I}_m$ for $m = 1, \ldots, k$. We hypothesize (3): These multiple "views" of $\mathcal{C}$ make $\mathcal{C}$ easier to learn in the sense of a

more favorable optimization landscape. This may be the key mechanism allowing task diversity to shorten the ICL plateau, and this phenomenon may extend beyond the ICL setup.

The following feature learning experiment makes this intuition more concrete. Consider the 2-layer feature learning setup, a setup with a large body of prior work (Damian et al., 2022; Ba et al., 2022; Dandi et al., 2023; Wang et al., 2023). For input $x \in \mathbb{R}^d$, the true function to learn is $f_\star(x) = U\sigma(A_\star x)$, where $U \in \mathbb{R}^{k\times h}$ and a true feature matrix $A_\star \in \mathbb{R}^{h\times d}$. The goal is to learn $f \approx f_\star$ with $f(x) = V\sigma(Wx)$, where $h' \gg h$, $V \in \mathbb{R}^{k\times h'}$, and $W \in \mathbb{R}^{h'\times d}$. The loss function is:

$$L(W,V) = \sum_{m=1}^{k} \mathbb{E}_{x\sim\mathcal{D}_x}\left[\left(v_i^\intercal \sigma(Wx) - u_i^\intercal \sigma(A_\star x)\right)^2\right], \quad U = [u_1 \quad \cdots \quad u_k]\,,\, V = [v_1 \quad \cdots \quad v_k]\,.$$

The idea is that $u_1, \ldots, u_k$ represent k sub-tasks that share the common feature matrix $A_\star$. If we sample $u_1, \ldots, u_k$ from k different distributions, this corresponds to multi-task training of k distinct tasks. Conversely, if we sample $u_1, \ldots, u_k$ from a single distribution, this corresponds to single-task training, as the k sub-tasks become identical. Interestingly, we find that multi-task training exhibits significantly shorter plateau compared to single-task training, as shown in Figure 7 (right). Figure 12 of the appendix shows that additional results with different hyperparameter configurations provide qualitatively similar results.

Although this toy model is a simple supervised learning setup, without sequence models or in-context learning, it reproduces the shortened plateau and makes our intuition more concrete through analogy. The results of this toy model, shown in Figure 7, lead us to make the general hypothesis (3): Training multiple tasks jointly with a shared structure makes it easier to learn the common structure.

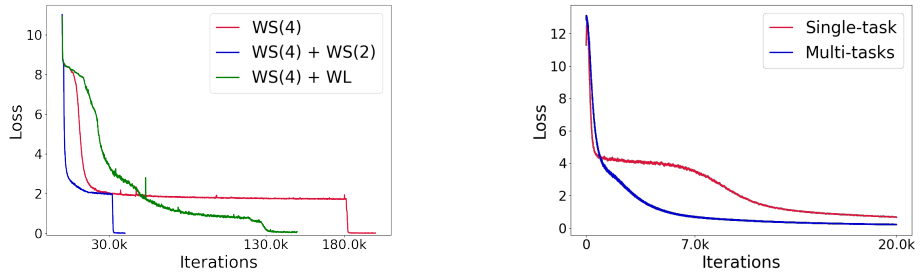


Figure 7: **(Left) Language ICL task**. Previous work Fu et al. (2024) identified the difficulty of learning the WordSelection(4) task. We found that mixing it with WordLength or WordSelection(2) reduces the plateau. Refer to Appendix C.1 for further details. **(Right) Feature learning setup**. For the toy model described in Section 4.2.2, multi-task feature learning significantly shortens the loss plateau. Refer to Appendix E for further details.

4.2.3 THE COMMON STRUCTURE IS NOT JUST AN INDUCTION HEAD

So then, what specifically is this common structure? We believe it must involve some algorithmic component, as all of the ICL tasks necessitate an internal algorithm to identify the specific function being demonstrated by the in-context demonstrations.

A plausible candidate is the *induction head*, a circuit that searches over the sequence for previous instances of a current token and predicts the same completion again. Indeed, Olsson et al. (2022) argued that the development of an induction head coincides with the escape from the training plateau. To test this idea, we designed the following Retrieval ICL task, inspired by the prior ICL tasks from Park et al. (2024); Singh et al. (2024); Reddy (2024).

Retrieval. Sample 1024 5-tuples of one key and four values: $\{(k_i, v_i^1, v_i^2, v_i^3, v_i^4)\}_{i=1}^{1024}$. The k_is and v_i^js are independently sampled from $\mathcal{D}_\mathcal{K}$ and $\mathcal{D}_\mathcal{V}$, respectively. To generate prompts, we uniformly sample $(n-1)$ 5-tuples without replacement and uniformly choose one v_i per 5-tuple, resulting in $\{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^{n-1}$. Next, we sample $p \sim \text{Unif}\left(\{1, \ldots, n-1\}\right)$ and set $\mathbf{q} = \mathbf{k}_p$. Finally, given $P_n = (\mathbf{k}_1, \mathbf{v}_1, \ldots, \mathbf{k}_{n-1}, \mathbf{v}_{n-1}, \mathbf{q})$, the task is to predict $\mathbf{v}_p$. We consider two Retrieval tasks: Gaussian Retrieval with $(\mathcal{D}_\mathcal{K}, \mathcal{D}_\mathcal{V}) = (\mathcal{N}(0, I_d), \mathcal{N}(0,1))$ and Boolean Retrieval with $(\mathcal{D}_\mathcal{K}, \mathcal{D}_\mathcal{V}) = (\text{Unif}\,\{\pm 1\}^d, \text{Unif}\,\{\pm 1\})$.

Note that the induction head is precisely the mechanism for solving this Retrieval task. We conduct a checkpoint experiment for the Retrieval tasks, allowing the models to learn the induction head. To train continuous ICL tasks, we use a checkpoint model pre-trained with Gaussian Retrieval. To train Boolean ICL tasks, we use a checkpoint model pre-trained with Boolean Retrieval. We find that the checkpoint does *not* significantly shorten the ICL tasks' plateaus. This result demonstrates that the common structure shared by other ICL tasks is not just an induction head. (It is unclear whether an induction head is useful for our ICL tasks at all.) For further details, please refer to Appendix D.2.

Presently, the problem of characterizing the common structure with any specificity remains unresolved. We defer further investigation of this matter to future work.

5 In-context learning tasks

In this section, we quickly list and define the ICL tasks that we consider, which are primarily adapted from Garg et al. (2022); Bhattamishra et al. (2024). Each ICL task is specified by $\mathcal{F}$ the function class, $\mathcal{D}_{\mathcal{F}}$ a probability distribution over the function class, and $\mathcal{D}_{\mathcal{X}}$ a probability distribution over the inputs. For continuous ICL tasks, $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, I_d)$. For boolean ICL tasks, $\mathcal{D}_{\mathcal{X}} = \mathrm{Unif}\,\{\pm 1\}^d$.

Continuous ICL tasks:

- **Linear Regression.** $\mathcal{F} = \{f \mid f(x) = w^\intercal x\}$. $\mathcal{D}_{\mathcal{F}}$: Each element of $w \in \mathbb{R}^d$ is independently sampled from $\mathcal{N}(\mu, 1)$.
- **Quadratic Regression.** $\mathcal{F} = \{f \mid f(x) = x^\intercal W x\}$. $\mathcal{D}_{\mathcal{F}}$: Each element of $W \in \mathbb{R}^{d \times d}$ is independently sampled from $\frac{1}{\sqrt{d}}\mathcal{N}(\mu, 1)$.
- **Sparse Linear Regression.** $\mathcal{F} = \{f \mid f(x) = w_{\text{sparse}}^\intercal x\}$. $\mathcal{D}_{\mathcal{F}}$: Each element of $w \in \mathbb{R}^d$ is independently sampled from $\mathcal{N}(\mu, 1)$. To sample w_{sparse}, we uniformly choose $k = 3$ coordinates and retain the corresponding coordinates of w.
- **ReLU Regression.** $\mathcal{F} = \{f \mid f(x) = \mathrm{ReLU}\,(w^\intercal x)\}$. $\mathcal{D}_{\mathcal{F}}$: Each elements of $w \in \mathbb{R}^d$ is independently sampled from $\mathcal{N}(\mu, 1)$.
- **Decision Tree.** Consider a full binary tree of fixed depth$= 4$. The values of leaf nodes and branch nodes are independently sampled from $\mathcal{N}(\mu, 1)$ and $\mathrm{Unif}\,(1, \ldots, d)$, respectively. When x traverses through the tree, at each branch node with index i, we move right if $x[i] > 0$ and move left otherwise. $f(x)$ corresponds to the value of the leaf node reached at the end of the traversal.

Boolean ICL tasks:

- **Sparse Parity**(k)**.** $\mathcal{F} = \{f \mid f(x) = \prod_{i \in A} x[i]\}$. $\mathcal{D}_{\mathcal{F}}$: $A \subseteq \{1, \ldots, d\}$ is a uniformly sampled subset of size k.
- **Parity.** $\mathcal{F} = \{f \mid f(x) = \prod_{i \in A} x[i]\}$. $\mathcal{D}_{\mathcal{F}}$: $A \subseteq \{1, \ldots, d\}$ is a uniformly sampled subset, regardless of the size.
- **Conjunction.** $\mathcal{F} = \{f \mid f(x) = (\wedge_{i \in A} x[i]) \wedge (\wedge_{i \in B} \bar{x}[i])\}$. $\mathcal{D}_{\mathcal{F}}$: A is an uniformly sampled subset of $\{1, \ldots, d\}$. Thereafter, we uniformly sample a subset $B \subseteq \{1, \ldots, d\} \backslash A$. .
- **Disjunction.** $\mathcal{F} = \{f \mid f(x) = (\vee_{i \in A} x[i]) \wedge (\wedge_{i \in B} \bar{x}[i])\}$. $\mathcal{D}_{\mathcal{F}}$: A is an uniformly sampled subset of $\{1, \ldots, d\}$. Thereafter, we uniformly sample a subset $B \subseteq \{1, \ldots, d\} \backslash A$.

6 Conclusion

In this work, we identify that training on a diverse set of multiple ICL tasks is surprisingly easier than training for a single ICL task in the sense of a more favorable optimization landscape. This observation aligns with the "blessing of dimensionality/scale" seen in the modern era of deep learning. Indeed, LLM training via next-token prediction can be thought of effectively as a highly diverse multi-task learning, requiring a wide range of reasoning skills for a wide range of text types, and the success of LLM training may be attributed not only to the richness of the data at scale but also to the easier optimization (training) induced by the diversity of natural language training data.

This insight opens new avenues for future work. It may be that explaining and understanding the success of large-scale deep learning requires considering not just the large data, large network, and large compute but also the large (effective) task diversity.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *NeurIPS*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *ICLR*, 2023.

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. *ICML*, 2024.

Jimmy Ba, Murat A. Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *NeurIPS*, 2022.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *NeurIPS*, 2023.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12: 149–198, 2000.

Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *Foundations of Computational Mathematics*, 2024.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and llms by learning to learn discrete functions. *ICLR*, 2024.

Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *NeurIPS*, 2022.

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

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *NeurIPS*, 2022.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality. *COLT*, 2024.

Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. *ICML*, 2024.

Yuan Cheng, Songtao Feng, Jing Yang, Hong Zhang, and Yingbin Liang. Provable benefit of multi-task representation learning in reinforcement learning. *NeurIPS*, 2022.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. *ICML*, 2021.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv 2009.09796*, 2020.

Alex Damian, Jason D. Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. *COLT*, 2022.

Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. How two-layer neural networks learn, one (giant) step at a time. *Mathematics of Modern Machine Learning Workshop at NeurIPS 2023*, 2023.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. *arXiv 2301.00234*, 2024.

Simon Shaolei Du, Wei Hu, Sham M. Kakade, Jason D. Lee, and Qi Lei. Few-shot learning via learning the representation, provably. *ICML*, 2021.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv 2310.17086*, 2023.

Jingwen Fu, Tao Yang, Yuwang Wang, Yan Lu, and Nanning Zheng. Breaking through the learning plateaus of in-context learning in transformer. *ICML*, 2024.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *NeurIPS*, 2022.

Nikhil Ghosh, Song Mei, and Bin Yu. The three stages of learning dynamics in high-dimensional kernel methods. *ICLR*, 2022.

Angeliki Giannou, Liu Yang, Tianhao Wang, Dimitris Papailiopoulos, and Jason D. Lee. How well can transformers emulate in-context newton's method? *arXiv 2403.03183*, 2024.

Riccardo Grazzi, Julien Niklas Siems, Simon Schrodi, Thomas Brox, and Frank Hutter. Is mamba capable of in-context learning? *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv 2312.00752*, 2024.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv 2306.11644*, 2023.

Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, and Yu Bai. How do transformers learn in-context beyond simple functions? a case study on learning with representations. *ICLR*, 2024.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. *arXiv 2406.02550*, 2024.

Jiachen Hu, Xiaoyu Chen, Chi Jin, Lihong Li, and Liwei Wang. Near-optimal representation learning for linear bandits and linear rl. *ICML*, 2021.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. *ICML*, 2020.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. Provably efficient reinforcement learning with linear function approximation. *COLT*, 2020.

Jikai Jin, Zhiyuan Li, Kaifeng Lyu, Simon S. Du, and Jason D. Lee. Understanding incremental learning of gradient descent: A fine-grained analysis of matrix sensing. *ICML*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arxiv 2212.04458*, 2024.

Yingcong Li, Xupeng Wei, Haonan Zhao, and Taigao Ma. Can mamba in-context learn task mixtures? *ICML 2024 Workshop on In-Context Learning*, 2024.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *ICLR*, 2024.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. *ICML*, 2024.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv 2101.06804*, 2021.

Rui Lu, Gao Huang, and Simon S. Du. On the power of multitask representation learning in linear mdp. *NeurIPS*, 2022.

Arvind V. Mahankali, Tatsunori Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *ICLR*, 2024.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *JMLR*, 17(81):1–32, 2016.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. *ACL*, 2022a.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *EMNLP*, 2022b.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *NeurIPS*, 2019.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. Distinguishing antonyms and synonyms in a pattern-based neural network. *EACL*, 2017.

Feng Nie, Meixi Chen, Zhirui Zhang, and Xu Cheng. Improving few-shot performance of language models via nearest neighbor calibration. *ICML*, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *arXiv 2209.11895*, 2022.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *ICML*, 2024.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. *ICML*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Allan Raventos, Mansheej Paul, Feng Chen, and Surya Ganguli. The effects of pretraining task diversity on in-context learning of ridge regression. *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *NeurIPS*, 2023.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. *ICLR*, 2024.

Maria Refinetti, Alessandro Ingrosso, and Sebastian Goldt. Neural networks trained with sgd learn distributions of increasing complexity. *ICML*, 2023.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *ACL*, 2022.

Jirko Rubruck, Jan P. Bauer, Andrew Saxe, and Christopher Summerfield. Early learning of the optimal constant solution in neural networks and humans. *arXiv 2406.17467*, 2024.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv 1706.05098*, 2017.

Brihat Sharma, Yanjun Gao, Timothy Miller, Matthew M. Churpek, Majid Afshar, and Dmitriy Dligach. Multi-task training with in-domain language models for diagnostic reasoning. *arXiv 2306.04551*, 2023.

Aaditya K. Singh, Ted Moskovitz, Felix Hill, Stephanie C. Y. Chan, and Andrew M. Saxe. What needs to go right for an induction head? a mechanistic study of in-context learning circuits and their formation. *ICML*, 2024.

Jiajun Song, Zhuoyan Xu, and Yiqiao Zhong. Out-of-distribution generalization via composition: a lens through induction heads in transformers. *arXiv 2408.09503*, 2024.

Linfeng Song, Kun Xu, Yue Zhang, Jianshu Chen, and Dong Yu. ZPR2: Joint zero pronoun recovery and resolution using multi-task learning and BERT. *ACL*, 2020.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *TMLR*, 2023.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *ICLR*, 2024.

Nilesh Tripuraneni, Michael I. Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. *NeurIPS*, 2020.

Nilesh Tripuraneni, Chi Jin, and Michael I. Jordan. Provable meta-learning of linear representations. *ICML*, 2021.

Nilesh Tripuraneni, Lyric Doshi, and Steve Yadlowsky. Can transformers in-context learn task mixtures? *NeurIPS Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *ICML*, 2023.

Johannes von Oswald, Eyvind Niklasson, Maximilian Schlegel, Seijin Kobayashi, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Blaise Agüera y Arcas, Max Vladymyrov, Razvan Pascanu, and João Sacramento. Uncovering mesa-optimization algorithms in transformers. *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Mingze Wang and Chao Ma. Understanding multi-phase optimization dynamics and rich nonlinear behaviors of relu networks. *NeurIPS*, 2023.

Yifei Wang, Yuyang Wu, Zeming Wei, Stefanie Jegelka, and Yisen Wang. A theoretical understanding of self-correction through in-context alignment. *arXiv 2405.18634*, 2024.

Zhichao Wang, Andrew Engel, Anand Sarwate, Ioana Dumitriu, and Tony Chiang. Spectral evolution and invariance in linear-width neural networks. *NeurIPS*, 2023.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *TMLR*, 2022.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently. *arXiv 2303.03846*, 2023.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *ICLR*, 2022.

Jiaqi Yang, Wei Hu, Jason D. Lee, and Simon S. Du. Impact of representation learning in linear bandits. *ICLR*, 2021.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context. *JMLR*, 25(49):1–55, 2024.

Yi Zhang, Arturs Backurs, Sebastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with LEGO: A synthetic reasoning task. *arXiv 2206.04301*, 2023a.

Zhihan Zhang, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. *ACL*, 2023b.

Yulai Zhao, Jianshu Chen, and Simon S. Du. Blessing of class diversity in pre-training. *AISTATS*, 2023.

Wenjie Zhou, Minghua Zhang, and Yunfang Wu. Multi-task learning with language modeling for question generation. *ACL*, 2019.

A EXPERIMENTAL DETAILS

Model Architectures. We focus on transformer (Vaswani et al., 2017), Mamba (Gu & Dao, 2024), and Hyena (Poli et al., 2023). To handle model architectures that process inputs and outputs as vectors in an embedding space, we first pad each $f(x_i)$ with $(d-1)$ zeros to match the dimension of x_is. We then add a learnable linear layer to map these vectors into the embedding space. A second learnable linear layer maps the model's output to a scalar. Each model's hyperparameters are dictated as follows. For transformer, we use GPT2 (Radford et al., 2019), with 12 decoder layers of embedding size 256 with 8 heads and relative positional encoding. Hyena follows the exact configuration of transformer's setup for the relevant hyperparameters. On the other hand, Mamba employs 24 layers, doubling the number of layers to match the model parameter size as is standard for Mamba. All models contain input and output projection heads that map input to the hidden state and hidden state to output. To train a model from scratch, we randomly initialize the model parameters.

Configuration. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0001. The batch size $B = 64$ corresponds to each task. Regarding the choice of $\ell(\cdot,\cdot)$, we use mean-squared error loss for continuous ICL tasks and cross-entropy loss for boolean ICL tasks. The input dimensions considered are $d = 10$ and 15.

Plateau and Training Exit Conditions. The plateau escape time t_{plateau} is defined as $t_{\text{plateau}} = \min_{t>100}\left[\frac{1}{100}\sum_{t'=t-99}^{t}\text{Training loss}(t') < 0.8\right]$. This effectively captures the plateau escape as we have normalized each task's loss around 1. The training exit conditions are defined as follows $t_{\text{exit}} = \min_{t>100}\left[\frac{1}{100}\sum_{t'=t-99}^{t}\text{Test error}(t') < 0.2\right]$ for continuous ICL tasks and $t_{\text{exit}} = \min_{t>100}\left[\frac{1}{100}\sum_{t'=t-99}^{t}\text{Test accuracy}(t') > 0.95\right]$ for boolean ICL tasks. t_{plateau} and t_{exit} are measured for each individual task.

Batch generation process The following pseudo algorithm summarizes the batch generation process of (multi)-task ICL training.

Algorithm 1 Batch generation process

Require: Multi-tasks $\{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$
Require: Corresponding domains $\{\mathcal{D}_{\mathcal{X}_1}, \ldots, \mathcal{D}_{\mathcal{X}_k}\}$
Require: Batch size for each task $\{B_1, \ldots, B_k\}$
Ensure: Generated batch of prompts

```
Initialize Batch B ← ∅
for m = 1 to k do
    for j = 1 to B_m do
        Sample function f ~ D_{F_m}
        Sample inputs x_1, ..., x_n ~^IID D_{X_m}
        Generate prompt: P^{(j,m)} = (x_1, f(x_1), ..., x_{n-1}, f(x_{n-1}), x_n, f(x_n))
        Update batch B ← B ∪ {P^{(j,m)}}
    end for
end for
```

After constructing a batch, we minimize the following empirical loss. Denote $f^{(j,m)}$ and $x_i^{(j,m)}$ as the corresponding in-context function and i-th input for each $P^{(j,m)}$.

$$\hat{L}(\theta) \colon = \sum_{m=1}^{k} c_m \left[\sum_{j=1}^{B}\left[\frac{1}{n}\sum_{i=1}^{n}\ell\big(M_\theta(P_i^{(j,m)}), f^{(j,m)}(x_i^{(j,m)})\big)\right]\right]$$

B ADDITIONAL TABLES

To demonstrate the generality of our claim, we conducted extensive experiments beyond those shown in Table 1. The following tables summarize results for transformer (input dimension $d = 15$), Mamba (input dimension $d = 10$), and Hyena (input dimension $d = 10$). We report two metrics for each run: time to escape the plateau and the time to complete the learning (written in parentheses).

Number of tasks	Boolean Tasks				Continuous Tasks				
	Sparse Parity(2)	Sparse Parity(3)	Conjunction	Disjunction	Linear Regression	Quadratic Regression	ReLU Regression	Sparse Linear Regression	Decision Tree
1	> 1000k	> 1000k	46.9k (66.6k)	22.6k (50.7k)	58.8k (61.3k)	25.0k (25.1k)	4.1k (5.4k)	31.6k (35.3k)	32.4k (86.2k)
2	38.9k (52.6k)	104.7k (148.2k)	7.5k (16.5k)	5.0k (11.6k)	12.4k (13.9k)	14.8k (17.2k)	4.1k (5.1k)	4.8k (5.4k)	26.6k (70.4k)
8	4.0k (4.8k)	10.1k (13.1k)	2.8k (8.5k)	2.7k (8.7k)	2.3k (3.7k)	4.0k (4.3k)	2.4k (3.9k)	2.2k (3.6k)	13.1k (44.4k)
9	3.6k (4.1k)	10.9k (11.6k)	2.5k (7.9k)	2.5k (7.8k)	2.0k (3.2k)	3.5k (3.6k)	2.0k (3.4k)	1.9k (3.2k)	13.8k (50.0k)

Table 2: **Task diversity shortens the ICL plateau.** We train a transformer with various combinations of 9 different tasks ($d = 15$).

Number of tasks	Boolean Tasks				Continuous Tasks			
	Sparse Parity(2)	Sparse Parity(3)	Conjunction	Disjunction	Linear Regression	Quadratic Regression	ReLU Regression	Sparse Linear Regression
1	7.5k (7.7k)	194.3k (196.5k)	0.4k (0.4k)	0.4k (0.4k)	0.2k (0.4k)	17.7k (37.7k)	0.2k (1.0k)	0.2k (0.4k)
2	2.9k (3.2k)	14.4k (15.6k)	0.4k (0.5k)	0.3k (0.4k)	0.2k (0.5k)	7.7k (22.8k)	0.3k (0.9k)	0.2k (0.5k)
4	1.5k (1.7k)	6.0k (6.7k)	0.3k (0.4k)	0.3k (0.4k)	0.2k (0.6k)	3.4k (17.1k)	0.3k (0.9k)	0.2k (0.6k)
8	2.4k (2.4k)	3.7k (4.3k)	0.3k (0.4k)	0.3k (0.4k)	0.3k (0.7k)	2.8k (20.5k)	0.3k (0.9k)	0.3k (0.6k)

Table 3: **Task diversity shortens the ICL plateau (Mamba).** We train a Mamba with various combinations of 9 different tasks ($d = 10$).

Number of tasks	Boolean Tasks				Continuous Tasks			
	Sparse Parity(2)	Sparse Parity(3)	Conjunction	Disjunction	Linear Regression	Quadratic Regression	ReLU Regression	Sparse Linear Regression
1	1.1k (1.3k)	53.9k (58.6k)	0.2k (0.3k)	0.2k (0.3k)	0.1k (0.4k)	11.1k (35.4k)	0.2k (0.7k)	0.1k (0.4k)
2	2.0k (2.4k)	23.7k (27.2k)	0.2k (0.3k)	0.2k (0.3k)	0.1k (0.4k)	7.6k (33.6k)	0.2k (0.7k)	0.1k (0.4k)
4	2.4k (2.5k)	16.8k (17.5k)	0.2k (0.3k)	0.2k (0.3k)	0.1k (0.5k)	6.3k (41.7k)	0.2k (0.8k)	0.1k (0.5k)
8	3.8k (4.0k)	13.6k (14.2k)	0.2k (0.3k)	0.2k (0.3k)	0.2k (0.5k)	7.1k (55.0k)	0.2k (0.9k)	0.2k (0.5k)

Table 4: **Task diversity shortens the ICL plateau (Hyena).** We train a Hyena with various combinations of 9 different tasks ($d = 10$).

For Mamba and Hyena, loss plateaus are less consistently observed. Specifically, training on Conjunciton, Disjunction, Linear regression, ReLU Regression, and Sparse Linear Regression rarely exhibit a loss plateau. This tendency, however, does not violate our claim, as we focus on how task diversity shortens the plateau when it *does occur*.

C LANGUAGE ICL TASKS

C.1 NATURAL LANGUAGE ICL TASKS

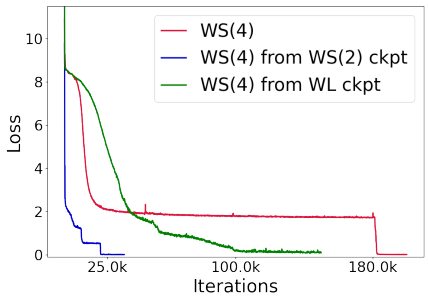


Figure 8: Checkpoint experiments on language ICL tasks.

To further verify our claim on tasks with real-world language data, we examine ICL in natural language processing (NLP) tasks.

Model. We use GPT2 (Radford et al., 2019) model with 12 layers, 12 attention heads, 768 embedding size, and 50257 vocabularies. We train GPT2 from scratch to learn in-context tasks.

Dataset. We use the word data from Nguyen et al. (2017). Following the approach in Todd et al. (2024), we choose 5245 words that can be tokenized to a single token. Words are then randomly chosen and arranged into sequences according to the task. Each sequence consists of six examples where the preceding five serve as context examples, and the last one as the query example.

Prompting. We convert a given sequence of words to a prompt by following Todd et al. (2024), where words are separated by a space, and examples are separated by a line:

<bos>Q. word$^1_1$ word$^1_2$ ... word^1_d
A. answer^1

$\vdots$

Q. word$^5_1$ word$^5_2$ ... word^5_d
A. answer^5

Q. word$^6_1$ word$^6_2$ ... word^6_d
A.

WordSelection(d)**.** We devise WordSelection(d) by generalizing the WordSelection Task (Fu et al., 2024). The goal of this task is to select a single word from given d words. The model's goal is to learn in-context which word to select.

WordLength. We also devise WordLength Task. Let len(w) be the length of word w and $\%$ be the remainder operator. Given four words $w_1, \dots w_4$, the goal of this task is to learn in-context either $\sum_{i=1}^{4}$ len(w_i) or $\sum_{i=1}^{4}$ len$(w_i)\%10$ from given examples.

Result. Prior work Fu et al. (2024) reported the difficulty of learning WordSelection(4), by showing that it cannot learn within about 78k iterations. Indeed, as shown in (Figure 7, Right), single-task training of WordSelection(4) exhibits a plateau for over 180k iterations. However, mixing WordSelection(4) and WordSelection(2) significantly shortens the plateau to 3.2k iterations, accelerating the learning. The same phenomenon is observed when WordSelection(4) and WordLength are trained simultaneously. These results reveal that our claim generally holds in NLP tasks.

We also conduct the checkpoint experiment, where we use *checkpoint* models on WordSelection(2) and WordLength as baseline models and train them respectively on WordSelection(4). As shown in Figure 8, using *checkpoint* models shortens the plateau, demonstrating that NLP tasks also share a certain common structure.

C.2 Formal Language ICL task: Regbench

We follow the configuration of Akyürek et al. (2024). To jointly train Quadratic Regression and DFA tasks, since their input formats differ, we use two different input embedding layers. Figure 9 illustrates the result.

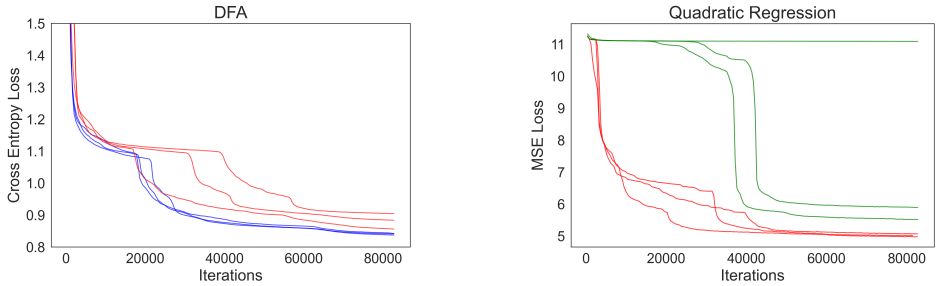


Figure 9: **Regbench.** DFA task shortens the Quadratic Regression's plateau, while Quadratic Regression does not shorten the DFA's plateau. Red line shows the multi-task training dynamics and blue and green lines show the single-task training dynamics.

D Checkpoint experiment

D.1 Experimental details

In this section, we provide the configurations for the checkpoint experiments in Section 4.2.1. The input dimension is set to $d = 10$. For each single-task training, we save the model once it escapes the plateau. The exact iteration numbers where the models are saved are provided in Table 5. Since Sparse Parity(2) and Sparse Parity(3) do not escape the plateau within observable training steps, no checkpoint models could be obtained. After saving the baseline models, we train each task (Conjunction, Disjunction, Linear Regression, Quadratic Regression, ReLU Regression, Sparse Linear Regression, Sparse Parity(2), and Sparse Parity(3)) according to the configuration in Appendix A. To calculate the ratio, we divide the result by the plateau escape time summarized in the first row of Table 1.

	Boolean Task			Continuous Task				
	Conjunction	Disjunction	Boolean Retrieval	Linear Regression	Quadratic Regression	ReLU Regression	Sparse Linear Regression	Gaussian Retrieval
Iteration number	2.2k	2.6k	105k	19k	11.7k	2.8k	7k	50k

Table 5: Checkpoint models.

Learning time comparison of checkpoint models. In Figure 6, we demonstrated that checkpoint models significantly shorten the plateaus. However, since the time required to escape the plateau and to learn the task may differ, it is necessary to verify whether the same phenomenon holds for learning time. Figure 10 confirms that this is indeed the case, although the effect is less robust compared to Figure 6.

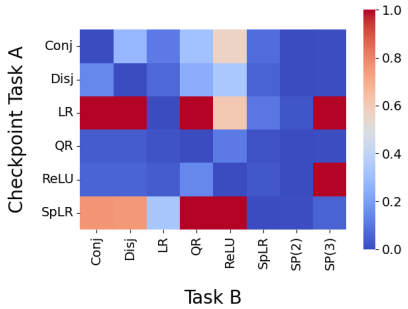


Figure 10: Learning time ratios of checkpoint models.

D.2 Experiment on Retrieval task

Recall the description of Retrieval task.

Retrieval. Sample 1024 5-tuples of one key and four values: $\{(k_i, v_i^1, v_i^2, v_i^3, v_i^4)\}_{i=1}^{1024}$. The k_is and v_i^js are independently sampled from $\mathcal{D}_{\mathcal{K}}$ and $\mathcal{D}_{\mathcal{V}}$, respectively. To generate prompts, we uniformly sample $(n-1)$ 5-tuples without replacement and uniformly choose one v_i per 5-tuple, resulting in $\{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^{n-1}$. Next, we sample $p \sim \text{Unif}(\{1, \ldots, n-1\})$ and set $\mathbf{q} = \mathbf{k}_p$. Finally, given $P_n = (\mathbf{k}_1, \mathbf{v}_1, \ldots, \mathbf{k}_{n-1}, \mathbf{v}_{n-1}, \mathbf{q})$, the task is to predict $\mathbf{v}_p$. We consider two Retrieval tasks: Gaussian Retrieval and Boolean Retrieval, with $(\mathcal{D}_{\mathcal{K}}, \mathcal{D}_{\mathcal{V}}) = (\mathcal{N}(0, I_d), \mathcal{N}(0, 1))$ and $(\mathcal{D}_{\mathcal{K}}, \mathcal{D}_{\mathcal{V}}) = (\text{Unif}\{\pm 1\}^d, \text{Unif}\{\pm 1\})$, respectively.

To train continuous ICL tasks, we use a checkpoint model saved from Gaussian Retrieval, and for boolean ICL tasks, we use a checkpoint model saved from Boolean Retrieval. We follow the configuration in Appendix A. For each experiment, we measure the time spent to learn each ICL task. The Results are illustrated in Figure 11.

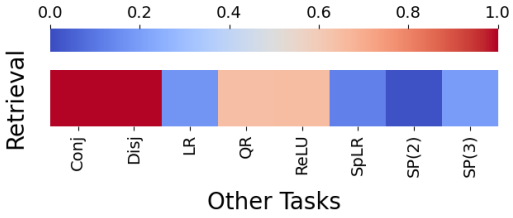


Figure 11: Checkpoint experiment on Retrieval.

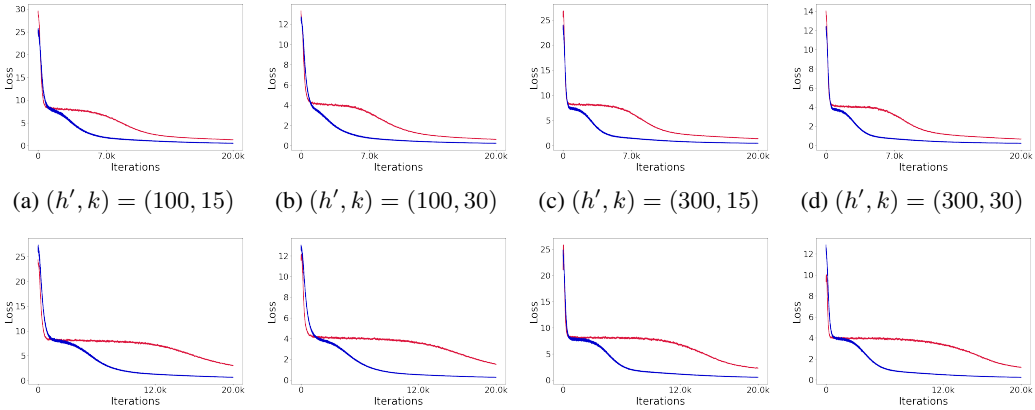
E Feature learning experiment

The true function is $f_\star(x) = U\sigma(A_\star x)$, where $U \in \mathbb{R}^{k \times h}$ and $A_\star \in \mathbb{R}^{h \times d}$. Columns of U, denoted as u_ms, are independently sampled from $\mathcal{N}(\mu_m, I_d)$, with $(\mu_1, \ldots, \mu_k)$ being pre-determined vectors prior to training.

To design multi-task learning, we independently sample k different vectors $\mu_1, \ldots, \mu_k$ from $\sqrt{h} \times \mathbb{S}^{h-1}$, making $\mathcal{N}(\mu_m, I_d)$s differ. In contrast, for single-task training, all $\mu_1, \ldots, \mu_k$ are set to the same vector μ, which is also sampled from $\sqrt{h} \times \mathbb{S}^{h-1}$.

The model that learns the true function is $f(x) = V\sigma(Wx)$ where $V \in \mathbb{R}^{m \times h'}$ and $W \in \mathbb{R}^{h' \times d}$. We set $d = 150$, $h \in \{10, 20\}$, $h' \in \{100, 300\}$, and $k \in \{15, 30\}$. The Adam optimizer is used with a learning rate of $\text{lr} = 0.001$ and we apply $0.1 \times \text{lr}$ for the model's heads, following the approach in Berthier et al. (2024). We use the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$.

Additional Figures. Figure 7 represents the result when $(d, h, h', k) = (150, 10, 100, 15)$. To enhance the generality, we present figures for other hyperparameters as well. For each figure, multi-task training (blue lines) exhibits a much shorter plateau compared to single-task training (red lines).



(a) $(h', k) = (100, 15)$ (b) $(h', k) = (100, 30)$ (c) $(h', k) = (300, 15)$ (d) $(h', k) = (300, 30)$

(e) $(h', k) = (100, 15)$ (f) $(h', k) = (100, 30)$ (g) $(h', k) = (300, 15)$ (h) $(h', k) = (300, 30)$

Figure 12: First row corresponds to the case when $(d, h) = (150, 10)$. Second row corresponds to the case when $(d, h) = (150, 20)$.

F No-context learning

F.1 Optimal no-context functions

In this section, we provide a detailed explanation of the optimal no-context function. We have defined the optimal no-context function as follows. For continuous ICL tasks, $g^\star_{\mathcal{F}} := \operatorname{argmin}_g \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}, x\sim\mathcal{D}_{\mathcal{X}}}\left[(g(x) - f(x)^2)\right]$, where the argmin is taken over $\{g \mid g\colon \mathbb{R}^d \to \mathbb{R}\}$. For boolean ICL tasks, $g^\star_{\mathcal{F}} := \operatorname{argmax}_g \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}, x\sim\mathcal{D}_{\mathcal{X}}}\left[\mathbf{1}(g(x) = f(x))\right]$, where the argmin is taken over $\{g \mid g\colon \{\pm 1\}^d \to \{\pm 1\}\}$.

For boolean ICL tasks, we can calculate $g^\star_{\mathcal{F}}(x)$ for each $x \in \{\pm 1\}^d$. The closed form of $g^\star_{\mathcal{F}}(x)$ can be expressed using $G_x := \{f(x) \mid f \in \mathcal{F}\}$ as follows.

$$g^\star_{\mathcal{F}}(x) = \begin{cases} 1 & \#(1 \in G_x) > \#(-1 \in G_x) \\ -1 & \text{(otherwise)} \end{cases}.$$

For continuous ICL tasks, we cannot apply the same approach since the output space is not discrete. However, we can still derive the closed form for $g^\star_{\mathcal{F}}(x)$ for most tasks: $g^\star_{\mathcal{F}}(x) := \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}}\left[f(x)\right]$. This is because

$$g^\star_{\mathcal{F}}(x) := \operatorname{argmin}_{a\in\mathbb{R}} \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}}\left[(f(x) - a)^2\right] = \operatorname{argmin}_{a\in\mathbb{R}} \left(a - \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}}[f(x)]\right)^2.$$

For tasks other than ReLU Regression, we can obtain the closed forms of $g^\star_{\mathcal{F}}(x)$ using the above property, as listed follows.

- Linear Regression: $g^\star_{\mathcal{F}}(x) = (\mu, \ldots, \mu)^\intercal x$.
- Quadratic Regression: $g^\star_{\mathcal{F}}(x) = x^\intercal U x, (U)_{ij} = \mu$.
- Sparse Linear Regression: $g^\star_{\mathcal{F}}(x) = \frac{k}{d}(\mu, \ldots, \mu)^\intercal x$.
- Decision Tree: $g^\star_{\mathcal{F}}(x) = \mu$.

For ReLU Regression, we can estimate $g^\star_{\mathcal{F}}(x)$ with small error through a sufficient number of samples from $\mathcal{D}_{\mathcal{F}}$.

F.2 EXPERIMENTAL DETAILS AND ADDITIONAL FIGURES

Let tasks with in-context function classes $\bigcup_{m=1}^{k} \mathcal{F}_m$ be trained simultaneously. Note that we do not discern the single-task and multi-task training: k can be 1 as well. During training, we measure

$$\mathbb{E}_{f \sim \mathcal{D}_{\mathcal{F}_m}, x_1, \ldots, x_n \overset{\text{IID}}{\sim} \mathcal{D}_{\mathcal{X}_m}} |M_\theta(P_n) - g^\star_{\mathcal{F}_m}(x_n)|^2, \quad P_n = (x_1, f(x_1), \ldots, x_{n-1}, f(x_{n-1}), x_n)$$

for continuous ICL tasks and

$$\mathbb{E}_{f \sim \mathcal{D}_{\mathcal{F}_m}, x_1, \ldots, x_n \overset{\text{IID}}{\sim} \mathcal{D}_{\mathcal{X}_m}} \left[\mathbf{1}(\text{sign}(M_\theta(P_n)) \neq g^\star_{\mathcal{F}_m}(x_n)) \right], \quad P_n = (x_1, f(x_1), \ldots, x_{n-1}, f(x_{n-1}), x_n).$$

for boolean ICL tasks. As shown by the following figures, these measurements are close to 0, which indicates that the model's output corresponds to *task-wise optimal no-context function*. We follow Appendix A for other configurations. Moreover, to prevent the optimal no-context be merely zero function, we use $\mu \neq 0$ to parameterize the in-context function classes.

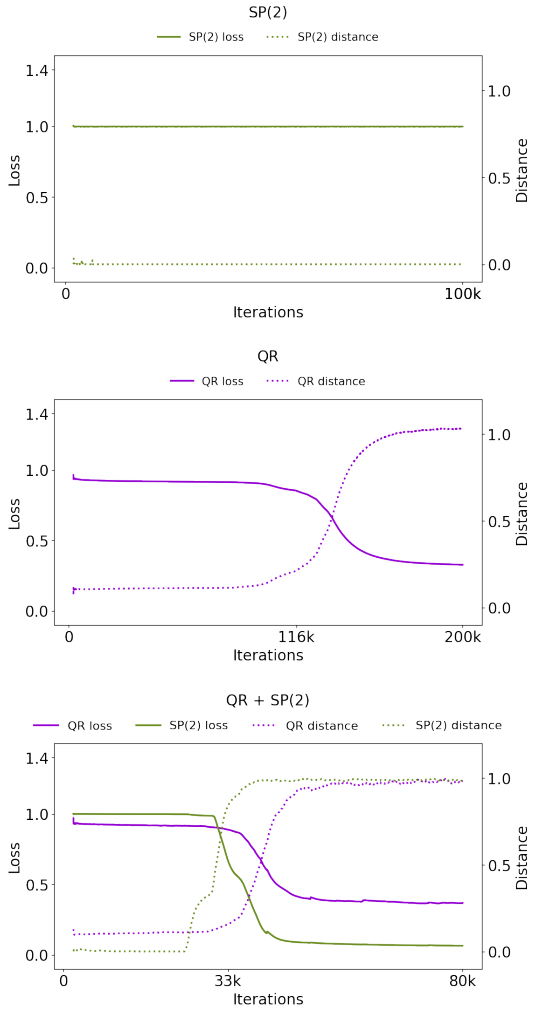


Figure 13: **No-context learning regime**. During the plateau, the model output matches the task-wise optimal no-context function. **(Top)** Sparse Parity(2). **(Middle)** Quadratic Regression ($\mu = -1$). **(Bottom)** Quadratic Regression ($\mu = -1$) + Sparse Parity(2).