Rationally Bayes: A Loss-Complexity Tradeoff Drives Algorithmic Strategies of In-Context Learning

Daniel Wurgaft[1*] **Ekdeep Singh Lubana**[2*] **Core Francisco Park**[2]
Hidenori Tanaka[2] **Gautam Reddy**[3] **Noah Goodman**[1,4]

[1]Department of Psychology, Stanford University
[2]CBS-NTT Program in Physics of Intelligence, Harvard University
[3]Joseph Henry Laboratories of Physics, Princeton University
[4]Department of Computer Science, Stanford University

Abstract

Recent work analyzing in-context learning (ICL) has identified a broad set of strategies that describe model behavior in different experimental conditions. We aim to unify these findings by asking *why* a model learns these disparate strategies in the first place. Specifically, we start with the observation that when trained to learn a mixture of tasks, as is popular in the literature, the strategies learned by a model for performing ICL can be encapsulated in a family of Bayesian predictors: a memorizing predictor, which assumes a discrete prior on the set of seen tasks, and a generalizing predictor, wherein the prior matches the underlying task distribution. This motivates a hierarchical Bayesian framework that describes a model's outputs via a posterior-weighted interpolation of the two predictors. To characterize the posterior probabilities, we take a rational analysis lens and, under reasonable assumptions about neural network learning dynamics, derive a predictive model that *almost perfectly* explains a trained neural network's outputs *without* assuming access to its weights. Crucially, this analysis makes explicit a *loss-complexity tradeoff* that helps explain why a model learns different strategies under varying experimental conditions: beyond how well it explains the data, a model's preference towards implementing a strategy is dictated by how "complex" it is. This interpretation helps explain several prior findings, while offering novel predictions: e.g., we show that sublinear sample efficiency during pretraining leads to a superlinear trend in the timescale for transience as task diversity is increased.

1 Introduction

In-Context Learning (ICL) has significantly expanded the open-ended nature of large language models (LLMs) [1–5], allowing them to learn novel behaviors from merely the provided context [6–12]. This has motivated a large body of work that analyzes controlled experimental settings to better understand ICL [13–17], leading to (i) behavioral accounts of *what* strategies are followed by a model to learn from its context [18–23], e.g., the ridge estimator in linear regression tasks [14, 17]; (ii) developmental accounts identifying *when*, i.e., under what training [24] and data [25] conditions, a particular strategy is used by the model [24–30]; or (iii) mechanistic accounts characterizing *how* such strategies get implemented [31–33], e.g., the use of induction heads [34]. While some have attempted characterizing ICL as a single procedure [22, 35, 36], more recent work has argued that the broader phenomenology of ICL stems from a model learning different strategies under varying experimental conditions [26, 30, 37].

These results suggest that a remaining hurdle for developing a unified account of ICL is understanding *why*, in the first place, models learn different strategies with disparate generalization abilities. Indeed,

*Equal contribution. Email: wurgaft@stanford.edu, ekdeeplubana@fas.harvard.edu.

Preprint. Under review.

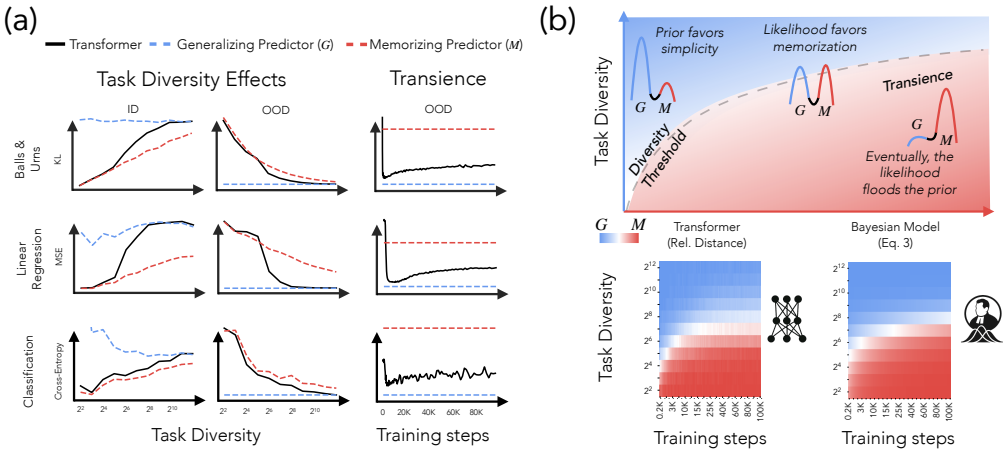

Figure 1: **Why Does a Model Learn Different Strategies for Performing ICL?** To answer this question, we analyze three distinct settings wherein a model is trained to learn a mixture of tasks. **(a) Model Behavior Transitions Between Memorizing and Generalizing Predictors.** We first make the observation that across settings, as diversity of data distribution and amount of training are increased, model behavior transitions between two Bayesian predictors: a memorizing predictor, M, which assumes a discrete prior over seen tasks, vs. a generalizing predictor, G, which assumes a continuous prior over the true distribution. This recapitulates prior results on task-diversity thresholds [25] and transient generalization [24] in a unifying language. **(b) A Hierarchical Bayesian Framework Provides an Explanatory and Predictive Account of ICL.** The consistency of these transitions motivates a hierarchical Bayesian model of ICL, wherein a model's inference-time behavior is framed as a posterior-weighted interpolation of the Bayesian predictors M and G, while pretraining is seen as a process of updating the preference (posterior probability) toward different predictors. We find that under reasonable assumptions regarding neural network learning dynamics, our framework is highly predictive of model behavior *throughout* training, *without* access to model weights (bottom panel). Additionally, our framework provides an explanatory account of ICL phenomena via a tradeoff between the *loss* and the *complexity* of learned solutions (top panel).

given that memorizing the training distribution almost always leads to better performance, why does the model learn an 'underfitting', out-of-distribution generalizing solution at all [24, 25]? Moreover, if capacity limitations prevent memorization, why does the model, among all underfitting solutions, learn the one that captures the *true* generative process [25, 26]? Finally, why does it *first* learn such a solution, only to eventually give way to one that does not generalize to novel tasks [24, 27]?

This work. To address the questions above, we first make the observation that in popularly studied ICL settings, where a model is trained to learn a mixture of tasks, identified strategies can be unified in the language of Bayesian inference: across three distinct settings, we show models learn solutions that behaviorally match Bayes-optimal strategies towards generalizing to the distribution of tasks seen during training vs. the true distribution from which said tasks are sampled (see Fig. 1(a)). These solutions respectively assume a discrete prior over the seen tasks (a *memorizing* predictor) or a continuous prior over the true distribution (a *generalizing* predictor), capturing previously studied setting-specific solutions (e.g., those described by Raventós et al. [25] in in-context linear regression). This observation then motivates our core contribution: we propose to understand ICL by invoking the approach of *rational analysis* from cognitive science [38–42], wherein a learner's behavior is explained as an *optimal* adaptation to data, given its objective and *computational constraints*. In our case, building on the finding that Transformers transition between memorizing and generalizing predictors, we examine how a Bayes-optimal learner would trade off between these solutions across training and data conditions, given a *simplicity* bias [43–47] and power-law scaling of loss with dataset size [48, 49]. This yields a *hierarchical* Bayesian framework that casts pretraining as a process of weighing preference (*posterior probability*) toward different solutions based on their *loss* and *complexity*. At inference, model behavior is framed as a posterior-weighted average of these solutions (which themselves are Bayesian—hence the term "hierarchical"). Deriving a closed-form expression for next-token predictions based on this framework, we are able to almost perfectly explain model behavior *throughout* training *without* assuming access to the weights. This allows us to both

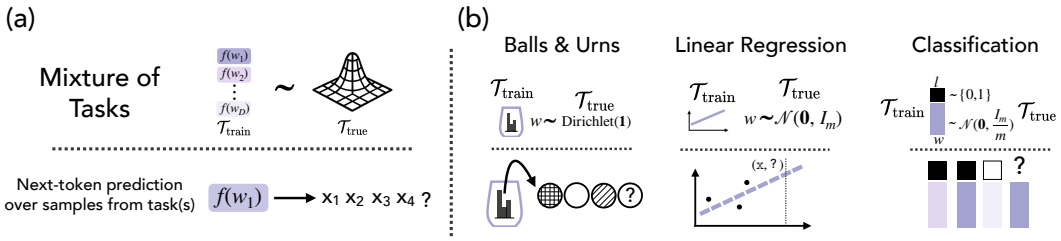

Figure 2: **Experimental Settings: Learning a Finite Mixture of Tasks. (a) General Formulation.** Popularly studied experimental settings in the literature on ICL can be seen as training a model to learn a distribution defined using a mixture of tasks (denoted $\mathcal{T}_{\text{train}}$), where each task is a parameterized latent function whose parameters are sampled from a distribution $\mathcal{T}_{\text{true}}$. **(b) Considered Settings.** We analyze three distinct instantiations of this general formulation: *Balls & Urns*, which captures the belief update interpretation of ICL and is a simplification of the Markov modeling setting from prior work [26, 52], and two popularly studied settings from the literature that capture the few-shot learning interpretation of ICL, i.e., in-context *linear regression* [14, 25] and *Classification* [20, 24, 28, 29].

explain several known ICL phenomena and draw novel predictions. Overall, we make the following contributions in this work.

- **Grounding a Hierarchical Bayesian Model of ICL in Rational Analysis.** We design a hierarchical Bayesian framework that models ICL as a *posterior predictive* over two Bayesian predictors—memorizing (discrete prior over seen tasks) vs. generalizing (continuous prior over the true distribution). In contrast to several previous Bayesian accounts of ICL [35, 50, 51], which focus solely on modeling Transformer behavior at inference-time as implementation of a single solution, our framework considers a *prior learned from pretraining* over multiple solutions—this decision turns out to be crucial for enabling a predictive account at both training and inference.
- **Loss-Complexity Tradeoff Explains Data-Diversity Effects and Transience.** Under reasonable assumptions about neural network learning dynamics (power-law scaling and simplicity bias), we derive a closed-form expression for log-posterior odds over the two predictors. This expression *almost perfectly* explains a neural network's behavior throughout training *without* assuming access to its weights. Crucially, it makes explicit a *tradeoff* between which predictor a model prefers depending on the predictor's loss on the training distribution and its complexity. This helps explain *transitions in model behavior*, as driven by increase in data-diversity [25] or training time [24], are in fact *implicitly* induced by changes in a predictor's complexity or loss.
- **Novel Predictions and Empirical Validation.** We validate our predictive account and the identified loss-complexity tradeoff over three broad experimental settings—Balls & Urns, Linear Regression, and Classification. Beyond explaining known phenomena, our account yields *novel predictions*, e.g., eliciting an infinite slowdown of transience as data-diversity is increased.

Overall, we argue our work advances a unifying explanatory and predictive account of ICL. More broadly, our results suggest the value of taking a *normative* perspective towards explaining neural networks, viewing learning behavior (both in-context and throughout training) as a *rational* adaptation to data properties and computational constraints.

2 Preliminaries: Learning a Finite Mixture of Tasks

To capture both few-shot learning [2, 14, 20, 25, 53] and belief update formulations [26, 34, 54, 55] of ICL, we analyze three distinct experimental settings in this paper. To this end, we first provide a general formulation of the learning setup, which allows us to formalize a unified language for examining model strategies for ICL in the next section.

General Formulation. We cast prior settings used for studying ICL [20, 25, 26] as learning a finite mixture of tasks. Specifically, settings analyzed in this work involve learning a mixture distribution $\mathcal{T}_{\text{train}}$ defined over D parametrized functions $\{f(\boldsymbol{w}_1, \cdot), \ldots, f(\boldsymbol{w}_D, \cdot)\}$, or 'tasks'. For each function $f(\boldsymbol{w})$, the parameters $\boldsymbol{w} \in \mathbb{R}^m$ are sampled from a predefined distribution $\mathcal{T}_{\text{true}}$ (see Fig. 2). We call D the *task diversity* of the mixture. Every training iteration, we randomly select a function $f(\boldsymbol{w}, \cdot) \in \mathcal{T}_{\text{train}}$, and use it to generate a sequence of length C (details vary by setting, see below). Batches consist of independently generated such sequences. We autoregressively train Transformers (GPT-NeoX architecture [56, 57]) for a predefined number of iterations N. Model performance is evaluated either on in-distribution (ID) sequences drawn from $\mathcal{T}_{\text{train}}$, or on out-of-distribution (OOD)

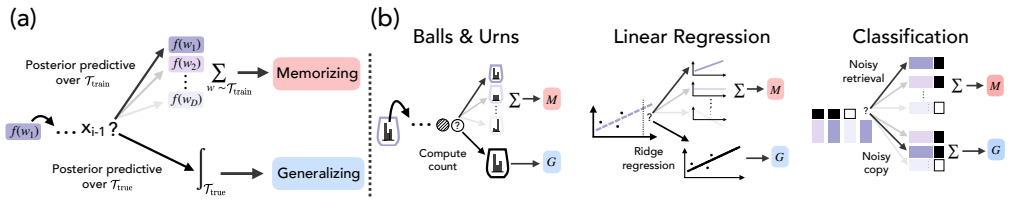

Figure 3: **Predictors in Different Experimental Settings. (a) Memorizing and Generalizing Predictors.** We compare model behavior to two idealized Bayesian predictors: (i) *Memorizing predictor* (M), which assumes a discrete prior over the mixture distribution $\mathcal{T}_{\text{train}}$, and (ii) *Generalizing predictor* (G), which assumes a prior over $\mathcal{T}_{\text{true}}$, the distribution from which tasks are sampled. **(b) Task-Specific Instantiations.** These predictors yield closed-form solutions (App. G); e.g., in Balls & Urns, the memorizing predictor computes a posterior-weighted average over urns seen in training, whereas the generalizing predictor uses empirical unigram statistics with pseudo-counts.

sequences drawn from the underlying distribution $\mathcal{T}_{\text{true}}$ (see App. F for further details on architecture and method). We analyze three specific instantiations of this general formulation, detailed next.

Balls & Urns. Related to the belief update formulation of ICL, this setting is inspired by the classic 'Urn Problem' from the probability literature [58]. Specifically, one draws (with replacement) balls from an urn containing balls of m types, and the goal is to estimate the distribution of ball types. Since solving this task only requires inferring unigram statistics of the input (a histogram), this setting simplifies the Markov modeling setup proposed by Park et al. [26]. A task $f(\boldsymbol{w}) = \text{Categorical}(\boldsymbol{w})$ denotes a stochastic map ('urn') from which states ('balls') are sampled, with $\boldsymbol{w} \sim \mathcal{T}_{\text{true}} = \text{Dirichlet}(\mathbf{1})$. Thus, the distribution $\mathcal{T}_{\text{train}}$ consists of D 'urns' $\{\text{Categorical}(\boldsymbol{w}_1), \ldots, \text{Categorical}(\boldsymbol{w}_D)\}$. To generate data, we sample C states from a randomly selected function $f(\boldsymbol{w})$, yielding a sequence $\boldsymbol{s} := [\boldsymbol{x}_1 \ldots \boldsymbol{x}_C]$, where $\boldsymbol{x} \sim \text{Categorical}(\boldsymbol{w})$.

Linear Regression. A standard problem setting in literature on understanding the few-shot learning formulation of ICL [14, 25, 53]. Here, the goal is to learn to in-context solve linear regression problems. A task $f(\boldsymbol{w}) = \boldsymbol{w}^\intercal \boldsymbol{x} + \epsilon$ denotes a noisy linear map that transforms a continuous input via a linear map $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_m)$ and introduces additive noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2)$, where $\boldsymbol{w} \sim \mathcal{T}_{\text{true}} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_m)$ and $\boldsymbol{I}_m$ denotes the $m \times m$ identity matrix. Thus, the training distribution $\mathcal{T}_{\text{train}}$ consists of D linear mappings $\{f(\boldsymbol{w}_1, \cdot), \ldots, f(\boldsymbol{w}_D, \cdot)\}$. To generate data, we sample C inputs and transform them with a randomly selected function $f(\boldsymbol{w}, \boldsymbol{x})$, yielding a sequence $\boldsymbol{s} := [\boldsymbol{x}_1, \boldsymbol{w}^\intercal \boldsymbol{x}_1 + \epsilon_1, \ldots, \boldsymbol{x}_C, \boldsymbol{w}^\intercal \boldsymbol{x}_C + \epsilon_C]$.

Binary Classification. Another popularly studied setting in literature on understanding the few-shot formulation of ICL [20, 28]. Our parameterization is inspired by Nguyen and Reddy [29], who define a task $f(\boldsymbol{w}, \boldsymbol{l}) = \boldsymbol{w} \oplus \boldsymbol{l}$ to denote an item-label pair, with $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{I}_m/m), \boldsymbol{l} \in \{0, 1\}$ defining $\mathcal{T}_{\text{true}}$. Thus, the training task distribution $\mathcal{T}_{\text{train}}$ consists of D item-label pairs $\{\boldsymbol{w}_1 \oplus \boldsymbol{l}_1, \ldots, \boldsymbol{w}_D \oplus \boldsymbol{l}_D\}$. Unlike other settings, multiple functions $f(\boldsymbol{w}, \boldsymbol{l})$ are used to generate data: first, $C - 1$ item-label pairs $\boldsymbol{w} \oplus \boldsymbol{l}$ are randomly sampled (with replacement) from $\mathcal{T}_{\text{train}}$. A pair is chosen from the sequence at random to be the query pair—it is appended to the end of the sequence, and its label is corrupted to be -1. We noise these items via $\tilde{\boldsymbol{w}} = \frac{\boldsymbol{w} + \sigma \boldsymbol{\epsilon}}{\sqrt{1+\sigma^2}}$, with $\sigma \in \mathbb{R}$ acting as the within-class variance, and $\boldsymbol{\epsilon} \in \mathcal{N}(0, \boldsymbol{I}_m/m)$. This process yields a sequence $\boldsymbol{s} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{C-1}, \boldsymbol{x}_{\text{query}}] = [\tilde{\boldsymbol{w}}_1 \oplus \boldsymbol{l}_1, \ldots, \tilde{\boldsymbol{w}}_{C-1} \oplus \boldsymbol{l}_{C-1}, \tilde{\boldsymbol{w}}_{\text{query}} \oplus -1]$. Models are only trained to predict the label of $\tilde{\boldsymbol{w}}_{\text{query}}$.

3 What Strategies: Memorizing and Generalizing Predictors

Our goal in this work is to understand *why* a model learns different strategies for performing ICL. We must thus first establish *what* these strategies are, allowing us to then characterize the dynamics driving changes in a model's preferred strategy. To this end, we build on the idea that autoregressive training with the next-token prediction objective corresponds to maximizing the likelihood of the data and learning the distribution underlying it [59]. We thus consider the two distributions forming the basis of our general formulation—i.e., $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{true}}$—and consider *optimal* strategies a learner can be expected to implement if it learns these distributions. This generalizes the approach of Raventós et al. [25], and yields the following two Bayesian predictors.

- **Memorizing Predictor (M).** The memorizing predictor assumes a discrete prior over the distribution of seen tasks ($\mathcal{T}_{\text{train}}$) and implements a posterior predictive of the form:

$$M(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) = \sum_{\boldsymbol{w}\sim\mathcal{T}_{\text{train}}} p(\boldsymbol{w}|\boldsymbol{s}_{1:i-1}) f_{\boldsymbol{w}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{T}_{\text{train}}}[f_{\boldsymbol{w}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1})]. \tag{1}$$

- **Generalizing Predictor (G).** The generalizing predictor assumes a continuous prior over the distribution from which tasks are sampled ($\mathcal{T}_{\text{true}}$), implementing a posterior predictive of the form:

$$G(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) = \int_{\boldsymbol{w}\sim\mathcal{T}_{\text{true}}} p(\boldsymbol{w}|\boldsymbol{s}_{1:i-1}) f_{\boldsymbol{w}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{T}_{\text{true}}}[f_{\boldsymbol{w}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1})]. \tag{2}$$

For all tasks we analyze, the predictors above can be defined in a closed-form manner (see App. G), mapping onto task-specific strategies defined in prior work: e.g., what has been called 'dMMSE' vs. ridge estimator in linear regression [25], 'in-weights learning' vs. 'in-context learning' in classification [20], and 'retrieval' vs. 'inference' in sequence modeling [26].

3.1 Validating the Memorizing and Generalizing Predictors

We next demonstrate the validity of the memorizing and generalizing predictors for the purpose of our analysis. Specifically, we show that as experimental conditions are varied, a model's behavior primarily transitions between these predictors. To this end, we consider two core phenomena associated with ICL—an increase in models' OOD performance with increasing *task diversity* D [25, 60, 61], and the 'forgetting' of this ability with increasing training steps N—a phenomenon known as *transient generalization* [24, 30, 62]. We first replicate these results behaviorally in Fig. 1(a), finding that, across settings, Transformers transition between performing like the memorizing predictor vs. like the generalizing predictor (see App. H for full results). Then, we make a direct comparison by computing the distance between our trained model's next-token predictions and the predictions of the memorizing and generalizing predictors. Specifically, let $d(.,.)$ denote a distance measure (KL-divergence or Euclidean distance), and denote the Transformer model trained from scratch via $h(.)$. Then, as a function of D and N, we can plot a heatmap of the *relative distance* between the trained model and the memorizing and generalizing predictors, defined as $d_{\text{rel}} = {}^{(r+1)}/_{2}$, where $r := \frac{d(h,G)-d(h,M)}{d(G,M)}$. This metric evaluates to 0 vs. 1 if the model is closer to the generalizing vs. the memorizing predictor. Results are shown in Fig. 4; see App. H.3 for absolute distances. We clearly see that increasing D for fixed N, the model first behaves like a memorizing predictor (in red), only to eventually transition to behaving like a generalizing predictor (in blue)—illustrating task-diversity effects [25]. Meanwhile, when increasing N for middle values of D, the model starts closer to a generalizing predictor, only to eventually give way to a memorizing predictor—illustrating transient generalization [24]. More broadly, across all tasks, we see there is a clear *delineation* of the model behavior into two regions of (N, D), such that model behavior is best explained by either the memorizing or the generalizing predictor in a given region. Given the optimality of these predictors on the distribution of seen tasks ($\mathcal{T}_{\text{train}}$) or the underlying distribution ($\mathcal{T}_{\text{true}}$), our analysis provides an explanation for *why* these predictors were observed in prior work. However, several questions remain, including, why is a generalizing strategy learned *even when it leads to worse ID performance*, and *why does varying experimental conditions change which strategy, among the memorizing and generalizing predictor, is implemented by a model*. We address these questions next.

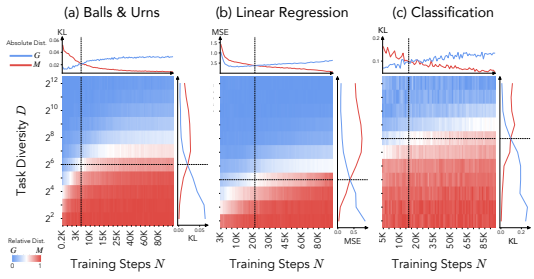

Figure 4: **Relative Distance Captures Transitions in Model Behavior.** We show the relative distance between model outputs and the two predictors. Marginals report the absolute distance values (e.g., KL between model and predictor outputs for the Balls & Urns setting), holding N constant (for the right plot), or D constant (for the top plot), and varying the other variable (denoted with the dotted line). Across all settings, we see model behavior decomposes into two regions explained by either the memorizing or generalizing predictor. In this figure, we use context length of 128, task dimensionality of 8 and MLP width of 256 for balls and urns. Linear regression has similar parameters except context length of 32, and classification has similar parameters other than MLP width of 512.

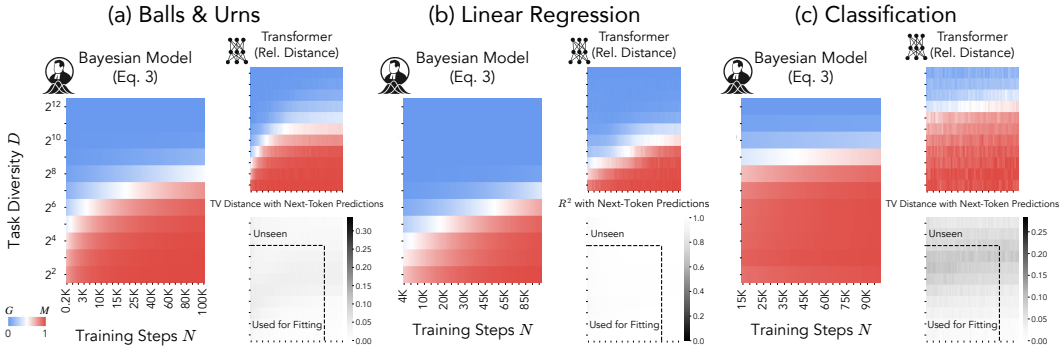

Figure 5: **Our Bayesian Model Captures Transitions Between Strategies Explaining Model Behavior.** We plot the posterior probability of the memorizing predictor given by our theoretical model (Eq. 4). Across three broad experimental settings—**(a)** Balls & Urns, **(b)** Linear Regression, and **(c)** Classification—we find our model identifies the regions best explained by a given predictor and the boundary between them, hence capturing the transition between solutions seen in a Transformer's training (as shown by the relative distance maps). Importantly, we find our model is highly predictive of the pretrained Transformer's behavior (next-token predictions), across both settings used for fitting the three free parameters of our model vs. unseen ones. Max color bar value for Balls & Urns and Classification is determined by the performance of a baseline predictor that always outputs the mean of the distribution $\mathcal{T}_{\text{true}}$. In this figure, we use context length of 256, task dimensionality of 8 and MLP width of 256 for balls and urns. Linear regression and classification have similar parameters except context length of 64 and 384, respectively.

4 Answering the Why: A Hierarchical Bayesian Account of ICL

Our analysis above shows that, except for intermediate values of N and D, model behavior is primarily explained by Bayes-optimal predictors capturing the distributions $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{true}}$. Motivated by these findings, we adopt the lens of *rational analysis* [38–42], a framework in cognitive science that aims to explain a learner's behavior as *optimal*, under *computational constraints*. What might be considered optimal in our case? Recall the fact that ICL is an *inductive* problem, i.e., a problem of predicting the next observation given past ones. Specifically, a predictor h^{pred} performing ICL predicts the i^{th} token $\boldsymbol{s}_i$ given previous elements in the sequence $\boldsymbol{s}_{1:i-1}$, using mechanisms it may have learned for this purpose based on sequences $S_{\mathcal{T}_{\text{train}}}(N, D)$ seen in training (denoted $S_{\mathcal{T}_{\text{train}}}$ from hereon for brevity). Then, given a hypothesis space of possible solutions the model has learned, Bayesian inference prescribes an optimal way to solve this problem via the *posterior predictive distribution*: compute a weighted average of predictions from each solution, with weights defined by a solution's posterior probability (i.e., a *posterior-weighted average*). Relying on the results of Sec. 3, we can assume our hypothesis space simply consists of the memorizing and generalizing predictors[2]. Thus, in our case, each solution itself corresponds to a Bayesian predictor—specifically, predictors M and G—hence resulting in the following *hierarchical Bayesian model*.

$$h^{\text{pred}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}, S_{\mathcal{T}_{\text{train}}}) = p(M|S_{\mathcal{T}_{\text{train}}})\; M(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) + p(G|S_{\mathcal{T}_{\text{train}}})\; G(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}). \tag{3}$$

The mathematical form of the predictor above frames in-context behavior as a *linear interpolation* in M and G, with posterior probabilities $p(M|S_{\mathcal{T}_{\text{train}}}), p(G|S_{\mathcal{T}_{\text{train}}})$, estimated from training, determining the interpolation weights. Then, in order to use this model to explain how a neural network performs ICL, we must estimate how posterior probabilities vary across training and data conditions.

Modeling the Posterior Probabilities. The posterior probability for a predictor Q, i.e., $p(Q|S_{\mathcal{T}_{\text{train}}}) \propto p(S_{\mathcal{T}_{\text{train}}}|Q)p(Q)$, is comprised of a likelihood term and a prior term—thus, these are the two terms we need to estimate. Inline with the perspective of rational analysis, in modeling these terms, we consider the following two well-known *computational constraints* of neural networks.

[2]While in principle it is possible that other predictors offer reasonable hypotheses for explaining model behavior, in the settings we analyze, we find that *very quickly* into training, other predictors (e.g., an optimal constant solution which always predicts the mean in linear regression [63, 64]) start to perform poorly compared to M and G in predicting model behavior. We thus focus our analysis on the regime where M and G are the primary hypotheses explaining model behavior (see App. E for further discussion).

- **A1:** Loss scales in a power-law manner with dataset-size N, i.e., $L(N) \approx L(\infty) + {}^{A}/_{N^{\alpha}}$, where $L(N)$ denotes the average loss on a dataset at time N and A is a constant that depends on model loss at initialization and training hyperparameters.
- **A2:** Neural networks exhibit a simplicity bias, such that the complexity of implementing a predictor Q follows a universal prior $p(Q) = 2^{-K(Q)^{\beta}}$, where $K(Q)$ denotes the estimated Kolmogorov complexity of Q. To accommodate the fact that we care about implementation in a Transformer, we instantiate a free parameter (β) on complexity while preserving the form of a universal prior.

A1 is merely a paraphrased version of well-known power-law scaling behaviors seen in neural network training [48, 49] and dictates how quickly observed data updates model behavior. That is, it offers a functional form for the rate at which likelihood in a posterior calculation grows, i.e., the rate of evidence accumulation. Meanwhile, **A2** is a well-established inductive bias of neural networks that has has found use in interpreting neural networks in past work [44, 59, 65]. This constraint dictates a model's preference over the two predictors (affecting the prior in a posterior calculation). The specific functional form we have chosen for modeling this constraint is inspired by the extensively studied notion of universal prior from algorithmic information theory, which defines the complexity of an algorithm as the description-length of the shortest program that implements it on a universal Turing machine [66–68]. In experiments, we follow standard practice in literature [69–71] and estimate an upper bound for this quantity by applying several compression algorithms to the code and data required for describing the predictor, and taking the minimal size in bits.

Returning to our goal, we now consider a model trained for N iterations on a task-mixture $\mathcal{T}_{\text{train}}$ of diversity D. The log-posterior odds of the two predictor can be defined as follows.

$$\eta(N, D) := \log \underbrace{\frac{P(M|S_{\mathcal{T}_{\text{train}}})}{P(G|S_{\mathcal{T}_{\text{train}}})}}_{\text{Posterior odds}} = \log \underbrace{\frac{P(S_{\mathcal{T}_{\text{train}}}|M)}{P(S_{\mathcal{T}_{\text{train}}}|G)}}_{\text{Bayes factor}} + \log \underbrace{\frac{P(M)}{P(G)}}_{\text{Prior odds}}.$$

Under constraints **A1**, **A2**, this simplifies as follows (see App. D).

$$\boxed{\eta(N, D) = \underbrace{\gamma N^{1-\alpha} \Delta L(D)}_{\text{Loss term}} - \underbrace{\Delta K(D)^{\beta}}_{\text{Complexity term}},} \tag{4}$$

where $\Delta K(D)^{\beta} := \log 2 \cdot (K_M(D)^{\beta} - K_G(D)^{\beta})$ is the difference between the exponentiated Kolmogorov complexity of the two predictors; $\Delta L(D) := L_G(S_{\mathcal{T}_{\text{train}}}(\infty, D)) - L_M(S_{\mathcal{T}_{\text{train}}}(\infty, D))$ is the difference between the average loss of the two predictors on a dataset of sequences sampled from $\mathcal{T}_{\text{train}}$; and γ is a constant related to the term A from constraint **A1**. To get the posterior probabilities for M and G, we simply convert η via the sigmoid function, denoted $\sigma(\cdot)$, yielding:

$$\boxed{h^{\text{pred}}(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) = \sigma\left(\eta(N, D)\right) M(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}) + \left(1 - \sigma\left(\eta(N, D)\right)\right) G(\boldsymbol{s}_i|\boldsymbol{s}_{1:i-1}).} \tag{5}$$

Note that the free parameters of this Bayesian model, i.e., (α, β, γ), depend on the problem setting and the Transformer's learning dynamics on it. To identify their values, we simply fit the Bayesian model's predictions to the pretrained Transformer $h(.)$'s next-token predictions on inputs retrieved from a subset of values (N, D). We emphasize that we *only fit three free parameters* across over 900 model checkpoints in 11 different training runs to get our results.

Validating the Model. We now check whether our model accurately captures the behavior of the pretrained Transformer and reproduces ICL's phenomenology. As shown in Fig. 5, our Bayesian model yields an *almost perfect prediction* of model next-token predictions for both seen / unseen settings. Moreover, without fitting to the relative distance maps, we find an *almost perfect match* between the posterior probabilities of the memorizing predictor given by our model and relative distance values. These results are replicated across *72 different maps* in App. H.4 with varying MLP width, context length and task dimensionality, yielding robust support for our model. Across all settings, we find our model is highly predictive of Transformer next-token predictions, with mean R^2 of 0.97 in linear regression, mean agreement of 0.92 in classification, and mean Spearman rank correlation of 0.97 in Balls & Urns. Additionally, we find very strong correlations of $0.99, 0.98, 0.99$ between our model's posterior probabilities and the relative distance values given by the Transformer in the Linear Regression, Classification, and Balls & Urns settings, respectively. Finally, we also examine ablations of our functional form in App. I, showing the computational constraints we assume are necessary to the success of our model.

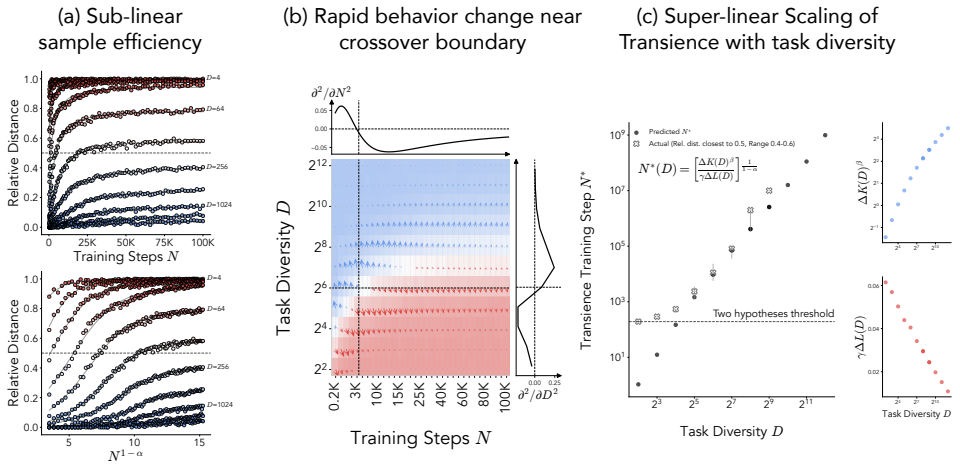

Figure 6: **Novel predictions from our framework. (a)** Our framework predicts that the posterior probability of the memorized predictor (and hence the relative distance, which can be thought of as an empirical estimate of this quantity) will show sub-linear scaling with respect to N, and sigmoidal growth with respect to $N^{1-\alpha}$ (holding D constant). **(b)** We predict a rapid change in model behavior for intermediate values of N and D, yielding a crossover-like boundary between the predictor that best explains model outputs. This can be seen via the magnitude of the second derivative of the relative distance near the boundary. Marginal plots show the second derivative of the relative distance with respect to N and D, with the data shown in these plots denoted on the vector map by the dotted lines. **(c)** Finally, our analysis predicts power-law scaling of the time of transience N^* (the time at which relative distance $d_{rel} = 0.5$) as diversity D increases, indicating the time to transience may become infinitely long amount with large D.

4.1 Novel Predictions

We next analyze our theoretical model to make informative qualitative predictions about Transformer behavior. Experiments hereafter involve the Balls & Urns setting. Unless stated otherwise, we use a context length of 128, task dimensionality of 8, and MLP width of 256.

- **Sub-linear growth from generalization to memorization.** Given the empirical match between the Bayesian model's posterior probabilities and the relative distance computed using the Transformer (see Fig. 6), we examine whether relative distance takes the functional form predicted by the model: sublinear growth with respect to N, and sigmoidal growth with respect to $N^{1-\alpha}$ (holding D constant). This arises from Eq. 4 and 5, in which evidence accumulates sub-linearly with the number of samples N (given $\alpha > 0$). As can be seen in Fig. 6(a), the relative distance of Transformer predictions between the memorizing and generalizing predictors clearly exhibits a sub-linear trend with N and sigmoidal growth with respect to $N^{1-\alpha}$, in accordance with the functional form predicted by Eq. 4. Note also that the bottom left panel of Fig. 6(a) clearly shows that even at high task diversity values, relative distance slowly increases towards the memorizing predictor in a sigmoidal manner. This stands in contrast to Raventós et al. [25]'s claim, including in their studied setting (see App. J), that at high D the Transformer only becomes *closer* to the generalizing predictor with increasing N.
- **Rapid behavior change near crossover.** At the point where the two hypotheses have equal log-posterior odds, our model predicts there will be a crossover in model behavior in terms of which predictor dominates the posterior. Specifically, the theoretically predicted boundary is defined as $\eta(N, D) = 0$. Given our sigmoidal functional form, we can also predict that behavior change will be rapid near this boundary, with small variations in experimental conditions (e.g., task diversity) yielding large changes in model behavior. To show this, we plot the second derivative of the relative distance in Fig. 6(b), and find that its magnitude is indeed greatest near the crossover boundary.
- **Scaling of time to Transience with Task-Diversity.** By equating $\eta(N, D) = 0$, our theory allows predicting the critical training time at which crossover from generalizing to memorizing occurs (i.e., relative distance $d_{\text{rel}} = 0.5$). Denoting this time to transience as N^*, we find it depends on task diversity D as follows: $N^*(D) = \left[\frac{\Delta K(D)^\beta}{\gamma \Delta L(D)}\right]^{\frac{1}{1-\alpha}}$. We thus plot the predicted time to transience as a function of D in Fig. 6(c), focusing on the regime where the Transformer has nontrivial windows of generalizing according to both predictors (since otherwise modeling transience does not make sense). As can be seen, our predictions hold really well with the empirically observed data; these

results improve further if we use a learning rate annealing schedule (see App. K). Importantly, we note that if this prediction derived from our model holds beyond our tested settings, it suggests that if the denominator in the expression becomes very small, e.g., when there are enough tasks in the mixture such that the memorizing and generalizing predictors produce similar outputs, the *time to transience can approach infinity*. In such a condition, generalization will persist *regardless* of how long a model is trained.

4.2 The Loss-Complexity Tradeoff

Having formalized and demonstrated the empirical validity of our hierarchical Bayesian framework, we now discuss the intuitive interpretation it offers us. Specifically, Eq. 4 suggests that the tradeoff between posterior-odds corresponding to different predictors is driven by the *loss* a predictor achieves on the training data and its *complexity* (see Fig. 7): early in training, the prior dominates, therefore a less complex solution—the generalizing predictor in our case—will be strongly favored as per the posterior calculation. However, the memorizing predictor will almost always have a lower loss than the generalizing predictor on training data. Thus, in low-to-medium task diversity settings, as training proceeds and N increases, the loss term in Eq. 4 will overtake the complexity term, i.e., the likelihood eventually dominates the posterior and 'floods' the prior. This will lead to the memorizing predictor becoming favored—explaining the transient generalization phenomenon from prior work [24]. In contrast, in high task diversity settings, the prior strongly disfavors the memorizing predictor due to its complexity, and given the sub-linear accumulation of likelihood (i.e., the $N^{1-\alpha}$ term), the time for transience to occur grows superlinearly—giving rise to the phenomenon of task diversity threshold seen in prior work [25]. The importance of a loss-complexity tradeoff for ICL was also arrived at by two recent papers examining training to perform ICL [30, 72], which we discuss further in Sec. 5.

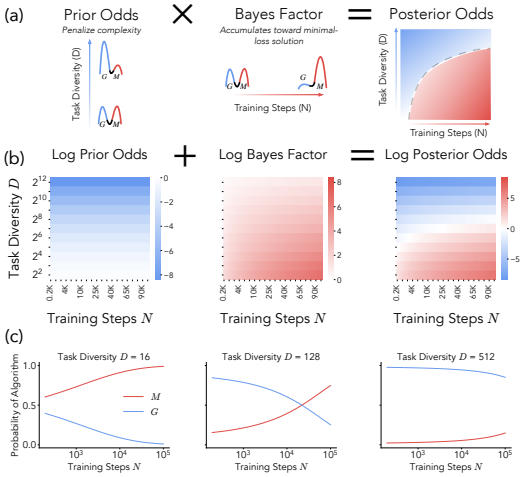

Figure 7: **Intuition Elicited by The Bayesian Model.** (a) Our framework suggests Transformers have a prior preference for learning simpler solutions, which often generalize better. However, throughout training, preference is updated towards solutions that better explain the data (i.e., have greater likelihood). This happens even at the expense of higher complexity, which in our case, yields a transition toward a memorizing predictor. (b, c) Our framework captures the tradeoff between solutions, showing that the *boundary* between the two regions corresponds to *equal posterior probabilities* of the two predictors.

To further exemplify the intuition above, we can consider variations on the training pipeline that can be expected to affect a model's ability to implement more complex predictors. Using the number of parameters as a control proxy to this end, we find that scaling MLP width raises the transition boundary between memorization and generalization—i.e., memorization is preferred more. *Our model closely captures this effect*: we see a decreasing complexity penalty β with larger MLP width, pointing toward the possibility that *larger models penalize complexity less*, yielding more memorization. Interestingly, we find that β decays exponentially with MLP width. Finally, we also vary context length and task dimensionality, and find that, once again, our model captures the effects of these interventions (see App. H.4).

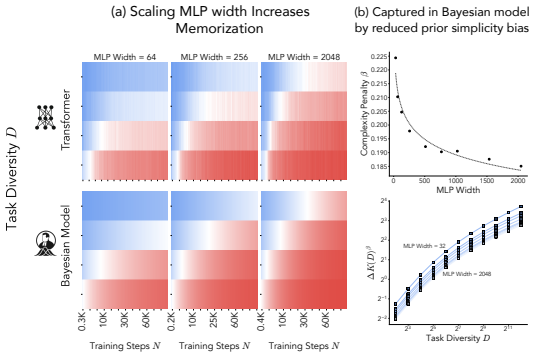

Figure 8: **Increased memorization with MLP width is captured by Bayesian model as reduced complexity penalty. (a)** We find relative distance to memorizing predictor decreases with MLP width. **(b)** This transition is closely captured by our Bayesian model, in which the 'complexity penalty' β, and hence the difference ΔK^{β}, decays exponentially with MLP width, yielding a reduced prior preference toward the generalizing predictor.

5 Discussion

In this work, we aim to unify findings from the ICL literature by asking *why* Transformers learn disparate strategies for performing ICL across varying training conditions. To do so, we take a *normative* perspective [38] which aims to explain Transformer learning as *optimal* under *computational constraints*. This lens yields a hierarchical Bayesian framework, which, assuming simplicity bias and scaling laws as computational constraints, offers a highly predictive account of model behavior: by fitting merely *three variables* to Transformer next-token predictions, we can *almost perfectly* predict model behavior across a spectrum of experimental settings *without* access to its weights. Our account also implies a fundamental trade-off that occurs throughout training between the *loss* and *complexity* of potential solutions learned by a model, with simpler, generalizing solutions learned early in training, while more complex, better-fitting solutions eventually becoming preferred. This tradeoff helps explain prior findings in the ICL literature, and provides novel predictions regarding training dynamics. Thus, we argue for our hierarchical Bayesian framework as an explanatory and predictive account of ICL, and a step towards a unified understanding of its phenomenology.

Relation to previous Bayesian models of ICL. Here, we discuss the relation between our work and other Bayesian or normative accounts of ICL. We provide an extended review of related work in App. B. Several prior works framed ICL as **Bayesian at inference-time**, meaning models implement a *single* Bayesian predictor in context [25, 35, 51, 62, 73, 74]. The focus on a single predictor in these works often led them to view ICL as a single strategy, with some focusing solely on a memorizing solution [35], while others refer mainly to a generalizing solution as ICL [25]. While there are cases in which in-context behavior is well approximated by a single predictor (e.g., the memorizing solution for low N, D), our results robustly show that several predictors are required to fully capture model behavior, and that the extent to which a particular predictor explains model behavior *varies* across training. Thus, a posterior-weighted average over *different* predictors, which considers a bias towards different predictors coming from training, rather than only inference-time, is required to *fully* capture model behavior across conditions. In contrast with studies focusing on inference-time, two recent works, Carroll et al. [30] and Elmoznino et al. [72], offer **Bayesian or normative views of pretraining to perform ICL**. Importantly, while these papers take different theoretical perspectives from ours, they arrive at a similar conclusion as us: the existence of a tradeoff between loss and solution complexity in pretraining. Elmoznino et al. [72] offer a normative theoretical analysis of training to perform ICL via next-token prediction loss, showing it yields an Occam's razor objective which minimizes loss as well as the learned solution's complexity. Carroll et al. [30] study task diversity effects and transient generalization in the linear regression setting of Raventós et al. [25]. Their Bayesian account of pretraining, which is rooted in theory of singular models [75] and makes different assumptions from ours, interestingly yields a relatively similar functional form for the posterior-odds. However, their measure of complexity is architecture-dependent [76], thus they only provide a qualitative analysis, as they cannot directly estimate the complexity of Bayesian predictors. In contrast, our hierarchical Bayesian framework provides a *quantitative*, predictive account of pretraining phenomena, in addition to capturing inference-time behavior as a weighted average of solutions, which is not addressed by Carroll et al. [30]. Despite that, we view Carroll et al. [30] and Elmoznino et al. [72]'s works as valuable contributions, in particular with regards to offering potential explanations for the source of the simplicity bias seen in Transformers.

Limitations. While we rely on a specific theoretical abstraction to arrive at our results, i.e., a hierarchical Bayesian model, we believe the predictive power of this abstraction corroborates its faithfulness. However, one limitation of our analysis is use of settings where model behavior is largely explained by only two predictors. It can be interesting to analyze settings where more predictors are feasible, e.g., the four predictors identified by Park et al. [26] in their task of learning a finite mixture of Markov chains. We believe this extension would be straightforward: we currently only characterize the dynamics of learning a memorizing vs. a generalizing predictor, but additionally modeling the statistics used to produce outputs (bigram vs. unigram) should accommodate the solutions identified by Park et al. [26]. Another important limitation of our analysis comes from the simple relation we assume between algorithmic complexity and complexity of implementation by a Transformer—while the assumption of simplicity bias is well-backed by theoretical and empirical claims [43–45], we believe our assumed relation could be improved by building on recent advances defining measures of how many effective parameters are used by a model to implement a solution [76, 77].

Takeaways. We see two main takeaways from our work for understanding of deep learning.

- **Is ICL Bayesian? Depends on Your Assumptions.** There exists debate regarding whether ICL can be viewed as Bayesian [25, 35, 50, 62]. Likely due to prior work largely focusing on ICL as a single strategy, some works that saw the emergence of a generalizing predictor have categorized it as 'non-Bayesian' [25, 62], since it is not the Bayes-optimal solution given the training distribution (which is the memorizing predictor). In contrast, taking the perspective of rational analysis, the right question is not *whether* ICL is Bayesian, but under *what assumptions* is it Bayesian. Clearly, the generalizing predictor is Bayes-optimal with respect to the true distribution $\mathcal{T}_{\text{true}}$. Moreover, as we show in this work, when taking into account a *simplicity bias* over predictors, learning a generalizing predictor *can* be considered Bayes-optimal in certain conditions. Thus, by extending modeling of ICL as Bayesian to both training *and* inference-time, we show that assessing the Bayes-optimality of ICL must consider the bias towards different predictors coming from pretraining. Cautiously, given our highly predictive results, we conclude that ICL can be considered as approximately Bayesian *given* constraints (assumptions) of a simplicity bias and sublinear sample efficiency (though see App. K for discussion of a deviation from Bayes-optimality that is not accommodated by our current assumptions).
- **The Value of a Normative Perspective.** An important takeaway that we believe may be of independent interest to the community is that, to understand generalization behavior in Transformers and neural networks more broadly, it may be enough to observe the structure of the data, and assume the network is well-approximated by a Bayes-optimal density estimator with a simplicity bias and sublinear sample efficiency. We hope our work shows that such a top-down *normative* perspective can provide highly predictive accounts, as well as provide potential explanations for *why* models behave the way they do, and we encourage the wider adoption of normative perspectives by the community.

Acknowledgments

We thank the Computation and Cognition Lab, in particular Ben Prystawski and Michael Li; Jay McClelland and the PDP group; the Physics of Intelligence group, especially Eric Bigelow; the CRISP lab at Harvard; Jesse Hoogland and Matthew Farrugia-Roberts; Surya Ganguli and the Neural Dynamics and Computation Lab; and Navin Goyal and the theory group at Microsoft Research India for useful discussions.

References

[1] OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model, 2023. URL https://arxiv.org/abs/2303.03378.

[4] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, et al. Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint arXiv:2404.09932*, 2024.

[5] OpenAI. Openai o3 and o4-mini system card, 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

[6] Gemini Team. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[7] AnthropicAI. Claude 3.7 sonnet system card, 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

[8] Keunwoo Peter Yu, Zheyuan Zhang, Fengyuan Hu, Shane Storks, and Joyce Chai. Eliciting in-context learning in vision-language models for videos through curated data distributional properties. *arXiv preprint arXiv:2311.17041*, 2023.

[9] Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-context learning enables robot action prediction in llms. *arXiv preprint arXiv:2410.12782*, 2024.

[10] Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. Iclr: In-context learning of representations. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Can Demircan, Tankred Saanum, Akshay K Jagadish, Marcel Binz, and Eric Schulz. Sparse autoencoders reveal temporal difference learning in large language models. *arXiv preprint arXiv:2410.01280*, 2024.

[12] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[13] 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, 2022. URL https://arxiv.org/abs/2209.11895.

[14] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

[15] Suraj Anand, Michael A Lepori, Jack Merullo, and Ellie Pavlick. Dual process learning: Controlling use of in-context vs. in-weights strategies with weight forgetting. *arXiv preprint arXiv:2406.00053*, 2024.

[16] Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms, 2024. URL https://arxiv.org/abs/2401.12973.

[17] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36, 2024.

[18] 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? *arXiv preprint arXiv:2202.12837*, 2022.

[19] 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, 2023. URL https://arxiv.org/abs/2303.03846.

[20] Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

[21] Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning, 2024. URL https://arxiv.org/abs/2402.18819.

[22] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[23] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International conference on machine learning*, pages 19565–19594. PMLR, 2023.

[24] Aaditya K. Singh, Stephanie C. Y. Chan, Ted Moskovitz, Erin Grant, Andrew M. Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers, 2023. URL https://arxiv.org/abs/2311.08360.

[25] Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. Competition dynamics shape algorithmic phases of in-context learning, 2024. URL https://arxiv.org/abs/2412.01003.

[27] Bryan Chan, Xinyi Chen, András György, and Dale Schuurmans. Toward understanding in-context vs. in-weight learning. *arXiv preprint arXiv:2410.23042*, 2024.

[28] Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task, 2023. URL https://arxiv.org/abs/2312.03002.

[29] Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning, 2024. URL https://arxiv.org/abs/2412.00104.

[30] Liam Carroll, Jesse Hoogland, Matthew Farrugia-Roberts, and Daniel Murfet. Dynamics of transient structure in in-context linear regression transformers. *arXiv preprint arXiv:2501.17745*, 2025.

[31] Kayo Yin and Jacob Steinhardt. Which attention heads matter for in-context learning? *arXiv preprint arXiv:2502.14010*, 2025.

[32] 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, 2024. URL https://arxiv.org/abs/2404.07129.

[33] Aaditya K Singh, Ted Moskovitz, Sara Dragutinovic, Felix Hill, Stephanie CY Chan, and Andrew M Saxe. Strategy coopetition explains the emergence and transience of in-context learning. *arXiv preprint arXiv:2503.05631*, 2025.

[34] Benjamin L. Edelman, Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. The evolution of statistical induction heads: In-context learning markov chains, 2024. URL https://arxiv.org/abs/2402.11004.

[35] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

[36] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

[37] Andrew Kyle Lampinen, Stephanie CY Chan, Aaditya K Singh, and Murray Shanahan. The broader spectrum of in-context learning. *arXiv preprint arXiv:2412.03782*, 2024.

[38] John R Anderson. *The adaptive character of thought*. Psychology Press, 2013.

[39] Nick Chater, Mike Oaksford, Nick Chater, and Mike Oaksford. Ten years of the rational analysis of cognition. *Trends in cognitive sciences*, 3(2):57–65, 1999.

[40] Thomas L Griffiths, Falk Lieder, and Noah D Goodman. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in cognitive science*, 7(2):217–229, 2015.

[41] Noah D Goodman, Joshua B Tenenbaum, Jacob Feldman, and Thomas L Griffiths. A rational analysis of rule-based concept learning. *Cognitive science*, 32(1):108–154, 2008.

[42] Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

[43] Preetum Nakkiran, Dimitris Kalimeris, Gal Kaplun, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Adv. in Neural Information Processing Systems (NeurIPS)*, 2019.

[44] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint. arXiv:1805.08522*, 2018.

[45] Feng Chen, Daniel Kunin, Atsushi Yamamura, and Surya Ganguli. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. *Advances in Neural Information Processing Systems*, 36:35027–35063, 2023.

[46] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *Adv. in Neural Information Processing Systems (NeurIPS)*, 2020.

[47] Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. Simplicity bias in transformers and their ability to learn sparse boolean functions. *arXiv preprint arXiv:2211.12316*, 2022.

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

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

[50] Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective. *arXiv preprint arXiv:2406.00793*, 2024.

[51] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023.

[52] Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pages 5793–5831. PMLR, 2022.

[53] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

[54] Eric J Bigelow, Ekdeep Singh Lubana, Robert P Dick, Hidenori Tanaka, and Tomer D Ullman. In-context learning dynamics with random binary sequences. *arXiv preprint arXiv:2310.17639*, 2023.

[55] Johannes A Schubert, Akshay K Jagadish, Marcel Binz, and Eric Schulz. In-context learning agents are asymmetric belief updaters. *arXiv preprint arXiv:2402.03969*, 2024.

[56] GPT-NeoX. Gpt-neox, huggingface., 2025. URL https://huggingface.co/docs/transformers/en/model_doc/gpt_neox.

[57] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022. URL https://arxiv.org/abs/2204.06745.

[58] Urn problem. In *Wikipedia*, December 2024. URL https://en.wikipedia.org/wiki/Urn_problem.

[59] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[60] 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 preprint arXiv:2406.02550*, 2024.

[61] Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

[62] Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism, 2024. URL https://arxiv.org/abs/2306.04891.

[63] Jirko Rubruck, Jan P. Bauer, Andrew Saxe, and Christopher Summerfield. Early learning of the optimal constant solution in neural networks and humans, 2024. URL https://arxiv.org/abs/2406.17467.

[64] Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy drives stagewise development in transformers, 2025. URL https://arxiv.org/abs/2402.02364.

[65] Jordi Grau-Moya, Tim Genewein, Marcus Hutter, Laurent Orseau, Grégoire Delétang, Elliot Catt, Anian Ruoss, Li Kevin Wenliang, Christopher Mattern, Matthew Aitchison, et al. Learning universal predictors. *arXiv preprint arXiv:2401.14953*, 2024.

[66] Marcus Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2005.

[67] Ray Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE transactions on Information Theory*, 24(4):422–432, 1978.

[68] R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7 (1):1–22, 1964. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(64)90223-2. URL https://www.sciencedirect.com/science/article/pii/S0019995864902232.

[69] Hector Zenil. A review of methods for estimating algorithmic complexity: Options, challenges, and new directions. *Entropy*, 22(6):612, 2020.

[70] Peter Grunwald and Paul Vitányi. Shannon information and kolmogorov complexity. *arXiv preprint cs/0410002*, 2004.

[71] Stephen Fenner and Lance Fortnow. Compression complexity. *arXiv preprint arXiv:1702.04779*, 2017.

[72] Eric Elmoznino, Tom Marty, Tejas Kasetty, Leo Gagnon, Sarthak Mittal, Mahan Fathi, Dhanya Sridhar, and Guillaume Lajoie. In-context learning and occam's razor, 2025. URL https://arxiv.org/abs/2410.14086.

[73] Aryaman Arora, Dan Jurafsky, Christopher Potts, and Noah D Goodman. Bayesian scaling laws for in-context learning. *arXiv preprint arXiv:2410.16531*, 2024.

[74] Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Explaining emergent in-context learning as kernel regression. *arXiv preprint arXiv:2305.12766*, 2023.

[75] Sumio Watanabe. A widely applicable bayesian information criterion. *The Journal of Machine Learning Research*, 14(1):867–897, 2013.

[76] Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and Susan Wei. The local learning coefficient: A singularity-aware complexity measure. *arXiv preprint arXiv:2308.12108*, 2023.

[77] Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. The developmental landscape of in-context learning, 2024. URL https://arxiv.org/abs/2402.02364.

[78] Yue M Lu, Mary I Letey, Jacob A Zavatone-Veth, Anindita Maiti, and Cengiz Pehlevan. Asymptotic theory of in-context learning by linear attention. *arXiv preprint arXiv:2405.11751*, 2024.

[79] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[80] Hong Jun Jeon, Jason D Lee, Qi Lei, and Benjamin Van Roy. An information-theoretic analysis of in-context learning. *arXiv preprint arXiv:2401.15530*, 2024.

[81] Rahul Ramesh, Mikail Khona, Robert P Dick, Hidenori Tanaka, and Ekdeep Singh Lubana. How capable can a transformer become? a study on synthetic, interpretable tasks. *arXiv preprint arXiv:2311.12997*, 2023.

[82] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

[83] Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650, 2023.

[84] Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 37:129696–129742, 2024.

[85] Tian Qin, Naomi Saphra, and David Alvarez-Melis. Sometimes i am a tree: Data drives unstable hierarchical generalization. *arXiv preprint arXiv:2412.04619*, 2024.

[86] Nigel Goldenfeld. *Lectures on phase transitions and the renormalization group*. CRC Press, 2018.

[87] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-6 (6):721–741, 1984.

Appendix

Table of Contents

A Glossary of Useful Terms

Generalizing Predictor (G). A predictor defined by a posterior-weighted average with a *continuous prior* over the true data-generating distribution $\mathcal{T}_{\text{true}}$. Such a predictor does not depend on the tasks seen during training, and can hence generalize well to novel, unseen tasks. This predictor maps onto 'task-learning' or 'inference' notions of ICL [26]. See also App. G for the precise mathematical form of generalizing predictors for settings studied in this work.

Memorizing Predictor (M). A predictor defined by a posterior-weighted average with a *discrete prior* defined over the distribution of tasks seen by the model during training, $\mathcal{T}_{train}$. Such a predictor depends on the tasks seen during training, and hence generalizes primarily to those tasks. This predictor maps onto 'task-retrieval' notions, though it also maps onto 'in-weights learning' in the case of the classification task introduced by Chan et al. [20]. See also App. G for the precise mathematical form of memorizing predictors for settings studied in this work.

Task diversity Threshold (D^*). For a given number of training steps N, task diversity threshold is defined as the critical number of tasks that must be involved in a training mixture such that the model behavior transitions from resembling a memorizing predictor to a generalizing predictor.

Transient Generalization (N^*). For a given task diversity D, when a model is trained for sufficiently long, its behavior transitions from more closely resembling a generalizing predictor to more closely resembling a memorizing one. We define this point as the point in which the relative distance between memorizing and generalizing crosses 0.5. This phenomenon is called transience, transient generalization, or the transient nature of in-context learning (see App. B for a longer discussion). The critical time to reach this point is denoted N^* in the main paper.

Relative Distance (r). A measure we use to characterize trade-off between predictors in settings where there are primarily two predictors (though one can easily generalize this measure to a setting with more predictors). Specifically, given a model h, two predictors Q_1, Q_2, and a distance function d, we define the term $r := \frac{d(h,Q_1)-d(h,Q_2)}{d(Q_1,Q_2)}$ and then relative distance as $d_{\text{rel}} = {}^{(r+1)}/_{2}$. The latter operation rescales r to a scale of 0 (if $h = Q_1$) to 1 (if $h = Q_2$), essentially assessing where the model h lives when a line is drawn with endpoints ranging from Q_1 to Q_2.

Posterior Odds, Prior Odds, Bayes Factor. Consider a set of observations X and two hypotheses H_1 and H_2 that are being assessed as candidates to explain the data. *Prior odds* are defined as the ratio $\frac{P(H_1)}{P(H_2)}$, i.e., if no observations are seen yet, which hypothesis is apriori preferred. *Bayes factor* is defined as the ratio of likelihoods $\frac{P(X|H_1)}{P(X|H_2)}$, i.e., once the observations are received, we compare how likely individual hypotheses deem these observations. Finally, *posterior odds* are defined as the ratio $\frac{P(H_1|X)}{P(H_2|X)} = \frac{P(H_1)}{P(H_2)} \times \frac{P(X|H_1)}{P(X|H_2)}$ = Prior odds $\times$ Bayes factor, i.e., how do observations affect the prior to help assess which hypothesis is more likely.

B Related Work

B.1 Prior Work Studying Task-Diversity Effects and Transience

We first discuss prior work studying phenomenology of ICL that forms the core of our paper: task diversity effects and transience. These works often focus on providing *mechanistic accounts* that are more bottom-up in nature, i.e., suitable for studying specific settings. For example, the induction head is a core mechanism employed by a model to perform the in-context classification task by Singh et al. [24], Reddy [28]. When transience occurs, the role of induction head is diminished, since it is not vital for implementing a memorizing predictor. However, as we show, transience occurs across a spectrum of settings, including ones where induction heads are never learned during training (e.g., Balls and Urns, which we analyze using a one-layer Transformer)—prior mechanistic accounts would not help justify transience in such settings. Our work thus takes a top-down account, i.e., we offer insights based on a *computational model* of ICL developed by capturing its phenomenology. This helps identify relevant control variables controlling a dynamic we argue lies at the heart of transience and task diversity effects: tradeoff between loss and complexity of a solution, which manifests itself into transitions between generalizing and memorizing predictors. Reconciling our computational model with mechanistic approaches undertaken in prior work can be an exciting avenue for progress.

- **Task diversity Threshold.** A phenomenon first popularized by Raventós et al. [25] in an in-context linear regression task, albeit originally demonstrated by Kirsch et al. [61] in a prior work on in-context classification of permuted MNIST images and recently expanded to a Markov modeling setting by Park et al. [26], to another classification setting by Nguyen and Reddy [29], as well as to a modular arithmetic setting by He et al. [60]. Specifically, Raventós et al. [25] empirically show that if one trains Transformers on a mixture of linear regression tasks, there is a critical number of tasks below which the model behavior is well-characterized by a Bayesian predictor over the seen tasks (what we term the memorizing predictor), while above it the behavior is well-characterized by the standard solution of ridge regression. These results were expanded in a recent theoretical work by Lu et al. [78], who study the asymptotic dynamics of a linear attention Transformer and show the change in solution used by the model as a function of task diversity is a second-order phase transition. In contrast to their work, our empirical analysis covers a broad range of settings that involve sequence modeling, regression, and classification, while the analytical parts of our paper provide a predictive framework that identifies the relevant control variables and explains how they affect the behavior of standard, nonlinear Transformers across all studied settings.
- **Transience / Transient Nature of In-Context Learning.** Originally observed by Chan et al. [20] while investigating the effects of data-centric properties on ICL, the term was popularized by Singh et al. [24]. Specifically, focusing on an in-context classification task, Singh et al. [24] showed that a model's ability to perform the generalizing ICL solution (employing a copy mechanism via the induction head) goes away when trained long enough. This phenomenon was recently generalized to a Markov modeling task by Park et al. [26] and to simplified variants of the in-context classification setting by Chan et al. [27], Nguyen and Reddy [29]. We especially emphasize the work of Nguyen and Reddy [29], who empirically identify a competition dynamic between a memorizing and generalizing solution for their task, building on this observation to perform a gradient flow analysis of their task and developing an effective theory of how transient generalization occurs. Our work differs in the sense that we provide an account of the competition dynamic and a model for its origins, instead of a gradient flow account that operates under the assumption of competition.

B.2 Prior Work Examining ICL from a Bayesian or Complexity-Based Lens

Different Bayesian and complexity-based accounts of ICL have been proposed in previous work. We clarify the innovation offered in our account and results compared to these prior studies.

- **Relation to Previous Bayesian Accounts of ICL**
  - **ICL as Bayesian at Inference Time.** Several prior works have modeled ICL as Bayesian at inference time, meaning they focused on models implementing a *single* Bayesian predictor in context, Raventós et al. [25], Xie et al. [35], Zhang et al. [51], Panwar et al. [62], Arora et al. [73], Han et al. [74]. For example, Xie et al. [35] use a setting involving mixture of Hidden Markov Models and argue ICL is akin to performing Bayesian averaging with a

discrete prior—what we call the memorizing predictor in our work. Due to the finite nature of their mixture, the authors never see the generalizing predictor emerge in their setting, i.e., a Bayesian averaging with a continuous prior over the true data-generating distribution. Even when some works saw the emergence of a generalizing solution, they categorized it as 'non-Bayesian' [25, 62], since it is not the Bayes-optimal solution for the inference problem — which is very often the memorizing solution. Our *hierarchical* perspective expands and generalizes the characterization of ICL provided in this literature. First, we show that ICL *cannot* be characterized as simply implementing a single Bayesian predictor, but must be thought of as a posterior-weighted average of different predictors, which is *updated* during training. In that sense, behaving as a generalizing solution *can* be Bayes-optimal, when taking into account a *simplicity prior* over predictors, in addition to the learning problem. Second, our framework extends modeling of ICL to both training *and* inference-time, showing that we must consider the bias towards different predictors coming from conditions of pretraining to understand ICL behavior at inference-time.

B.3 Broader Work on Understanding ICL and Meta-Learning

A crucial benefit of our Bayesian modeling framework is that one can retrieve posterior probabilities over the predictors the model predictions are being decomposed into. In other words, we can represent the model predictions as an interpolation of the predictors' outputs (since posterior probabilities sum to 1). Park et al. [26] propose a similar approach: "Linear Interpolation of Algorithms" (LIA). Specifically, LIA represents model predictions as an interpolation of the predictors' outputs, directly fitting the weights for interpolation. That is, for each condition of training time (N), task diversity (D) , LIA requires fitting a set of parameters, summing up to a total of $N \times D$ parameters fit (over 900 parameters per context-size and task dimensionality setting for our work). Meanwhile, our framework, by defining a precise functional form that explicates the role of N and D, minimizes the number of terms needed to perform fitting to 3—these terms are primarily related to model family, its learning dynamics, and intrinsic randomness of the data, hence making them hard to explicate. Crucially, one can see our framework as providing grounding to LIA: under the hood, LIA is trying to identify the posterior probabilities for all experimental settings! Moreover, our framework offers an explanation for *why* change in predictor weights occurs during training, and importantly, can *predict* dynamics of N, D conditions it was not trained on, which LIA cannot do since it has to be fit separately to each N, D condition.

Beyond the papers discussed above, there have been several complementary efforts to understand ICL from different perspectives (see the recent summary by Lampinen et al. [37] for a longer review). For example, several papers analogize ICL as implicitly performing optimization to learn novel tasks [22, 36, 53]; meta-learning in general [79, 80]; develop scaling laws for the sample efficiency of ICL [73]; identify limits of tasks that can be learned in-context [14, 17, 81]; demonstrate sudden learning curves exist in ICL [10, 54]; and characterize mechanisms employed by a model to perform ICL [13, 31, 34, 82]. Broadly, all these papers offer useful insights that are complementary to ours.

C Future Work

Our work also opens up several exciting avenues for further progress.

- **Can we Explain In-Context Transitions?** First, we note the phase transition elicited in this work primarily assumes a biased optimization process, and the ability to overcome this bias by seeing data supporting another solution. Accordingly, since ICL can be viewed as an optimization process [22, 83], it may be possible to use our results to explain behavioral transitions seen in recent work on in-context learning of novel concepts or behaviors [10, 54, 84].
- **Does our Framework Explain other Pretraining and ICL Phenomena?** we believe the competition dynamic characterized in our work is similar to the one demonstrated by Qin et al. [85] in a language modeling task. There, the authors show that depending on the diversity of rules of the language and training time, a model can either learn a bag of heuristics or the underlying grammar to generate sentences from the language. It may be possible to explain the phenomenology elicited in that work via the hierarchical Bayes lens we take in this paper, since there likely exists a tradeoff between compressibility and loss of a heuristic vs. grammar-learning solution.
- **Connecting our top-down framework to a bottom-up mechanistic account.** Our work intentionally takes a top-down approach, and hence does not offer a mechanistic account of how Transformer learning dynamics implement the loss-complexity tradeoff, or how the model weights different solutions at inference time. Such a bottom-up analysis likely requires studying either the gradient flow dynamics of ICL [29] or using mechanistic interpretability tools to examine circuits [32, 33].

D Derivations

Below, we derive formal expressions for the functional form of log-posterior odds (Eq. 4) and show how one can convert it into a predictive model (Eq. 5). For completeness, we repeat below the constraints underlying our modeling framework.

- **A1:** Loss scales in a power-law manner with dataset-size N, i.e., $L(N) \approx L(\infty) + {}^{A}/_{N^{\alpha}}$, where $L(N)$ denotes the average loss on a dataset at time N, and A is a constant that depends on model loss at initialization and training hyperparameters.
- **A2:** Neural networks exhibit a simplicity bias, such that the complexity of implementing a predictor Q follows a prior of the form $p(Q) = 2^{-K(Q)^{\beta}}$, where $K(Q)$ denotes the estimated Kolmogorov complexity of Q. To accommodate the fact that we care about implementation in a Transformer, we instantiate a free parameter (β) on complexity while preserving the form of a universal prior.

D.1 Log-Posterior Odds

We consider a parameterized model class $H(.)$ learning to implement a predictor $Q \in \{M, G\}$ when trained using a learner T on a dataset $S_{\mathcal{T}_{\text{train}}}(N, D)$ of N sequences sampled from the distribution $\mathcal{T}_{\text{train}}$ with diversity D. We approximate this model's learning dynamics via a hierarchical Bayes framework, i.e., we assume learning happens via a posterior update by computing likelihood of the data under all considered hypotheses (which are themselves Bayesian predictors, hence the term 'hierarchical'). Each hypothesis has an associated prior that reflects the learning pipeline's proclivity towards implementing it. For brevity, we will use the notation $S_{\mathcal{T}_{\text{train}}}$ to refer to the dataset, with sequences seen at update N denoted via a superscript $s^{(n)}_{\mathcal{T}_{\text{train}}}$; i.e., $S_{\mathcal{T}_{\text{train}}} = \cup_n S^{(n)}_{\mathcal{T}_{\text{train}}}$. We also use Θ_Q to denote the set of parameters in the landscape of model-class H, such that $H(\theta, \boldsymbol{s}) = Q(\boldsymbol{s})$ for any sequence $\boldsymbol{s}$ if $\theta \in \Theta_Q$.

We begin by analyzing the log-posterior of the predictor Q learned by the model:

$$
\begin{aligned}
\log P(Q|S_{\mathcal{T}_{\text{train}}}, T, H) &= \log P(\Theta_Q|S_{\mathcal{T}_{\text{train}}}, T, H) \\
&= \log \int_{\theta\in\Theta_Q} P(\theta|S_{\mathcal{T}_{\text{train}}}, T, H) \\
&\propto \log \int_{\theta\in\Theta_Q} P(S^{(N)}_{\mathcal{T}_{\text{train}}}|\theta)\, P(\theta|S^{(1)}_{\mathcal{T}_{\text{train}}}, \ldots, S^{(N-1)}_{\mathcal{T}_{\text{train}}}, T, H) \\
&\overset{\text{A1}}{=} \log \int_{\theta\in\Theta_Q} \prod_{n=1}^{N_{\text{eff}}} P(S^{(n)}_{\mathcal{T}_{\text{train}}}|\theta)\, P(\theta|T, H) \\
&= \log \int_{\theta\in\Theta_Q} \prod_{n=1}^{N_{\text{eff}}} P(S^{(n)}_{\mathcal{T}_{\text{train}}}|Q)\, P(\theta|T, H) \\
&= \sum_{n=1}^{N_{\text{eff}}} \log P(S^{(n)}_{\mathcal{T}_{\text{train}}}|Q) + \underbrace{\log \int_{\theta\in\Theta_Q} P(\theta|T, H)}_{\text{Prior of the learner and model-class towards learning } Q} \\
&\overset{\text{A2}}{=} N_{\text{eff}} \underbrace{\frac{1}{N_{\text{eff}}} \sum_{n=1}^{N_{\text{eff}}} \log P(S^{(n)}_{\mathcal{T}_{\text{train}}}|Q)}_{\text{Log likelihood of data under predictor } Q} - K(Q)^{\beta} \cdot \log_{\text{e}} 2 \\
&= N_{\text{eff}}\, L(Q) - K(Q)^{\beta} \cdot \log_{\text{e}} 2. \qquad (6)
\end{aligned}
$$

Overall, we have then,

$$\begin{aligned}\eta(N, D) &:= \log \frac{P(M|S_{\mathcal{T}_{\text{train}}}, T, H)}{P(G|S_{\mathcal{T}_{\text{train}}}, T, H)} \\ &= N_{\text{eff}} \left(\log\ P(S^{(n)}_{\mathcal{T}_{\text{train}}}|M) - \log\ P(S^{(n)}_{\mathcal{T}_{\text{train}}}|G)\right) + \left(-K(M)^{\beta} + K(G)^{\beta}\right) \cdot \log_{\text{e}} 2 \\ &= N_{\text{eff}}\ (L(G) - L(M)) - \left(K(M)^{\beta} - K(G)^{\beta}\right) \cdot \log_{\text{e}} 2 \\ &= N_{\text{eff}}\, \Delta L(D) - \Delta K(D)^{\beta}. \end{aligned} \tag{7}$$

In the above, our constraints get operationalized as follows.

- **A1** helps us accommodate the fact that while a Bayesian learner would make optimal use of all samples shown to it, neural network training in fact makes suboptimal use of samples seen during training, which we model by defining N_{eff}, i.e., the effective number of samples a neural network learns from. We will estimate this value below in Eq. 9.
- **A2** provides a form for the prior the learning pipeline (includes the learner T and model-class H) has towards implementing the predictor Q.

We next model N_{eff}. Specifically, we use the power-law scaling behavior of neural networks' learning dynamics to compute the loss reduced in N updates by such a pipeline, identifying the number of updates an idealized Bayesian learner would have to make in order to reduce loss by this amount.

$$\begin{aligned} N_{\text{eff}} &:= \frac{\text{Loss reduced under power-law scaling in } N \text{ updates}}{\text{Loss reduced by a Bayesian learner in a single update}} \\ &= \frac{\sum_{n=1}^{N} (L(n) - L(\infty))\delta n}{L(Q)} \\ &= \frac{1}{L(Q)} \sum_{n=1}^{N} \frac{A}{n^{\alpha}} \delta n \\ &= \frac{A}{L(Q)} N^{1-\alpha} \int_{0}^{1} \frac{1}{\hat{n}^{\alpha}} \delta \hat{n} \\ &= \gamma N^{1-\alpha}, \end{aligned}$$

where $\hat{n} = {}^{n}/_{N}$ and $\gamma = {}^{A}/_{L(Q)}$ is a constant that subsumes the loss of the predictor Q and the constant A from our assumed form of power-law scaling, which depends on the random loss of a network and effects of hyperparameters like batch-size and sequence lengths used to define the train data.

Substituting N_{eff} back into Eq. 7, we get our final model:

$$\boxed{\eta(N, D) = \gamma N^{1-\alpha} \Delta L(D) - \Delta K(D)^{\beta}.} \tag{8}$$

D.2 Converting from Posterior-Odds to a Predictive Model

At inference, the pretrained Transformer is shown a sequence $\boldsymbol{s}$, for which it makes a next-token prediction. To simulate this process in our framework, we define a Bayesian predictor, denoted h_{pred}, as follows.

$$\begin{aligned} h_{\text{pred}}(\boldsymbol{s}) &:= \sum_{Q \in \{M, G\}} P(Q|S_{\mathcal{T}_{\text{train}}}, T, H) Q(\boldsymbol{s}_C) \\ &= \sum_{Q \in \{M, G\}} \underbrace{P(Q|S_{\mathcal{T}_{\text{train}}}, T, H)}_{\text{Pretraining Prior}} \underbrace{Q(\boldsymbol{s})}_{\text{Prediction}} \\ &= \sum_{Q \in \{M, G\}} \frac{P(Q|S_{\mathcal{T}_{\text{train}}}, T, H)}{\sum_{Q \in \{M, G\}} P(Q|S_{\mathcal{T}_{\text{train}}}, T, H)} Q(\boldsymbol{s}) \\ &= \frac{\exp\left(\eta(N, D)\right)}{1 + \exp\left(\eta(N, D)\right)} M(\boldsymbol{s}) + \frac{1}{1 + \exp\left(\eta(N, D)\right)} G(\boldsymbol{s}). \end{aligned} \tag{9}$$

Using $\sigma(.)$ to denote the sigmoid function, we have the final form from Eq. 5 as follows.

$$\boxed{h_{\text{pred}}(\boldsymbol{s}) = \sigma(\eta(N, D))\, M(s) + (1 - \sigma(\eta(N, D)))\, G(s).} \tag{10}$$

Remark. It is worth highlighting that $\eta(N, D)$ essentially serves the role of free-energy in the analysis above. Use of free-energy to model an interpolation between two states of a system is a common theoretical framework used in physics to study systems that undergo transitions between a disordered state to an ordered state: e.g., see in Landau theory, one considers interpolations between free energy at high temperature and low temperature to model continuous (second-order) phase transitions [86]. Our overall theoretical model, and the phenomenology it elicits, are very similar to models from physics, a parallel that we believe can be worth pursuing in future work, e.g., to uncover universality behavior beyond what we considered in this paper.

E Two-Hypotheses Threshold: Minimum amount of training to enable the Hierarchical Bayesian Model

One can reasonably expect our proposed Hierarchical Bayesian model to explain learning dynamics of in-context learning will not be predictive of a Transformer's behavior early-on in training, for otherwise we are saying even an untrained model perfectly generalizes. In actuality, the Transformer becomes amenable to approximation by our model after some minimal amount of training has occurred. To automatically calculate whether we have finished this regime of training, we calculate an "optimal" interpolation between the two predictors if they were capable of explaining the model behavior: specifically, we rely on the relative distance as an estimation of the optimal interpolation weighting, and use it as a baseline to compare model outputs with. In particular, we compute the loss between this optimal interpolation baseline and our trained Transformer model, and if this loss is below a threshold, we claim our theoretical model is applicable. We call this threshold the **two-hypotheses threshold** (see Fig. 6).

To define the **two-hypotheses threshold**, we make the observation that while the loss between Transformer and interpolating predictor can be large to begin with, it very quickly reduces to a small value. We can expect this latter regime is where our theoretical model is most likely to accurate at modeling the trained Transformer's behavior. Motivated by this, we heuristically choose the two-hypotheses threshold as a loss value 20% higher than minimum for Balls & Urns and Classification, and 10% higher than minimum for Linear Regression, on the scale defined from minimum to maximum loss (we find that a stricter threshold is required for linear regression to .

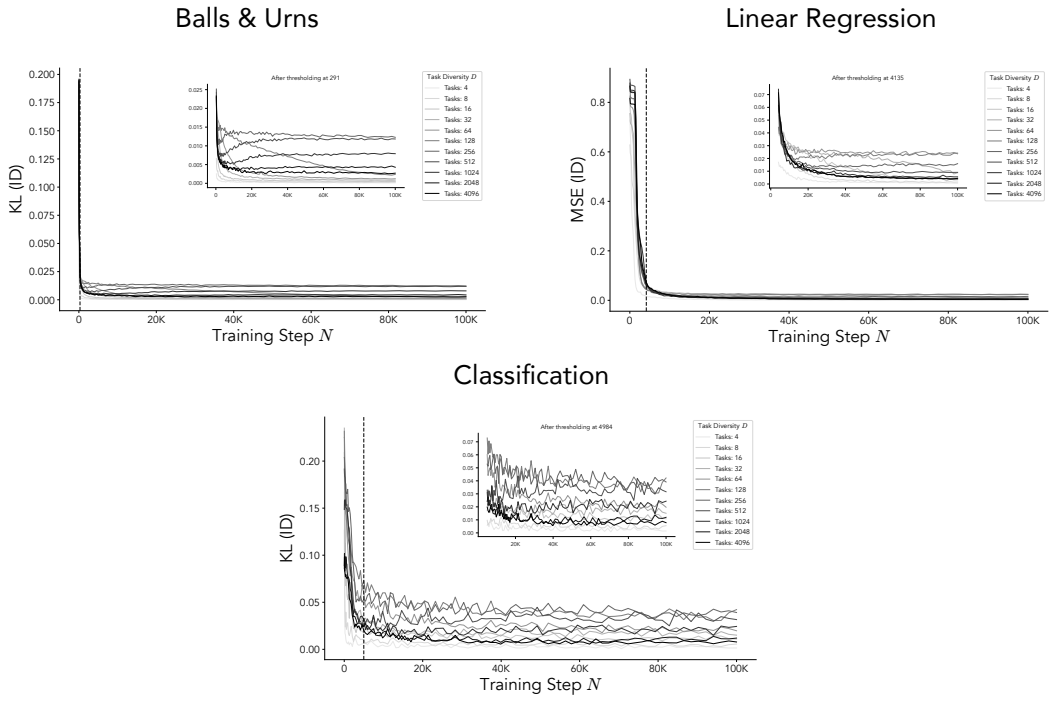

Figure 9: **Two-Hypotheses Threshold.** Defining an 'optimal' interpolation between the memorizing and generalizing predictors towards minimizing the Euclidean distance to the trained Transformer's predictions, we report the loss between this optimal interpolation and the Transformer's predictions. We observe a minimum amount of training is necessary for this loss to become sufficiently small such that our hierarchical Bayesian model, which implicitly assumes the Transformer can be functionally decomposed into the two predictors, will become applicable. The dotted lines demarcate this threshold, which we call the "two-hypotheses threshold". Mean-squared error (MSE) in the figure above is normalized by dimension. KL in this figure indicates forward KL from the Transformer's next token predictions to the interpolation of predictors.

F Experimental Details

F.1 Training and Model Details

Model. For all settings, we use the GPT-NeoX architecture sourced from Huggingface [56, 57]. While the number of layers / blocks in the model depend on the specific experimental setting (as reported below), we use only 1 attention head per layer and follow a sequential residual stream architecture across all settings.

Training. We use the Huggingface trainer with default parameters, changing only the learning rate, batch-size, total iterations, and warmup steps (reported below). Gradients are clipped to unit-norm. All models are trained on A100 GPUs, with maximum training budget reaching 2 days for all experiments encompassing the linear regression setting. We vary data-diversity D from $\{2^2, 2^4, \ldots, 2^{12}\}$ across all settings.

Settings-Specific Details. For our three core settings, we report results covering the following hyperparameters. We note that similar to Carroll et al. [30], as we vary task-diversity D, we include tasks from the lower diversity-values in the setting involving the larger one—this allows us to assess effects of increasing diversity on the learning of a given task.

- **Balls and Urns.** Models of hidden dimension size 64 are trained for 100K steps, with no warmup steps, at a constant learning rate of 5×10^{-4} and batch-size of 64. For our analysis, we derive experimental settings from combinations of task-dimensionality (equivalent to vocabulary-size), which varies in the set $\{8, 12, 16\}$; context length, which varies in the set $\{128, 256, 320\}$; and MLP expansion factor, which varies in the set $\{0.5, 4, 8\}$.
  - We conduct a separate experiment in which we attempt to elicit transience in higher task diversity settings ($D \in \{2^8, 2^9\}$). To do so on a reasonable compute budget (2M, 10M steps, respectively), we, similar to prior work [24, 29], have to intervene on the training pipeline. However, unlike prior work that often relies on weight decay for this purpose, we use learning rate annealing and find it to be sufficient. Specifically, we rely on an inverse square root schedule for decaying the learning rate with number of dimensions, context length, and MLP expansion fixed to 8, 128, and 4 respectively. We train up to $D = 2^7$ for 100K steps, and then train with $D = 2^8$ for 2M steps, and with $D = 2^9$ steps for 10M steps. Results of this experiment are shown in Fig. 6(c).
- **Linear Regression.** Models of hidden dimension size 64 are trained for 100K steps, with 5K warmup steps, at a constant learning rate of 5×10^{-4} and batch-size of 128. For our analysis, we derive experimental settings from combinations of task-dimensionality, which varies in the set $\{8, 12, 16\}$; context length, which varies in the set $\{16, 32, 64\}$; and MLP expansion factor, which varies in the set $\{0.5, 4, 8\}$.
- **Classification.** Models of hidden dimension size 64 are trained for 100K steps, with no warmup, at a constant learning rate of 5×10^{-4} and batch-size of 64. For our analysis, we derive experimental settings from combinations of task-dimensionality, which varies in the set $\{8, 16\}$; context length, which varies in the set $\{128, 256, 384\}$; and MLP expansion factor, which varies in the set $\{0.5, 4, 8\}$ for the 8 dimensions experiment, and is kept constant at 4 for the 16 dimensions experiment.

F.2 Analysis Details

Next, we specify broad details of our analysis pipeline. These notes clarify design decisions made in our evaluation and motivations underlying them.

General model and predictor evaluation. For **OOD evaluation** of both the Transformer and our procedurally defined predictors, i.e., the memorizing predictor M and generalizing predictor G, we draw 500 sequences from 500 *unseen* tasks (however, following still the same task distribution $\mathcal{T}_{\text{true}}$). In comparison, **ID evaluation** involves 500 sequences from *seen* tasks. If task-diversity D is less than 500, sequences from the same task may be seen multiple times. We note that the ID evaluation for classification is slightly different, following the popularly used pipeline of 'in-weights learning' first introduced by Chan et al. [20]: herein, one specifically ensures that a copy of the test item does

not appear in the context, thus disallowing the use of an in-context copying solution and testing memorization more explicitly.

Computing absolute distance between Transformer and predictors. Given an input, we use both the Transformer model and the procedurally defined predictors to make next-token predictions. Then, we compare distance between these predictions using either the symmetrized KL (average of forward and backward KL) for the Balls and Urns and Classification settings, or the mean-squared error (MSE) for linear regression.

Computing relative distance between Transformer and predictors. Recall that relative distance, for a given distance measure $d(.,.)$ between two functions or distributions (e.g., KL-divergence or Euclidean distance), is defined as $d_{\text{rel}} = {}^{(r+1)}/_{2}$, where $r := \frac{d(h,G)-d(h,M)}{d(G,M)}$ and $h(.)$ denotes the Transformer model trained from scratch. This metric implicitly makes the assumption that in some function space, the model $h(.)$ lies on a line between the predictors M and G. Correspondingly, for the scenarios this assumption is violated, the value of d_{rel} can go outside the range 0–1. This relatively rarely, but nevertheless noticeably, occurs (e.g., if the model implements the optimal constant solution early on in training for linear regression). Accordingly, we clamp the metric between 0–1.

Minimum amount of training to enable the Hierarchical Bayesian Model. One can reasonably expect our proposed Hierarchical Bayesian model to explain learning dynamics of in-context learning will not be predictive of a Transformer's behavior early-on in training, for otherwise we are saying even an untrained model perfectly generalizes. In actuality, the Transformer becomes amenable to approximation by our model after some minimal amount of training has occurred. To automatically calculate whether we have finished this regime of training, we calculate an "optimal" interpolation between the two predictors if they were capable of explaining the model behavior: specifically, if the euclidean distance had to be minimum between the interpolation of predictors and Transformer outputs, we can identify a closed-form solution for the interpolation weights and use it as a baseline to compare model outputs with. In particular, we compute the loss between this optimal interpolation baseline and our trained Transformer model, and if this loss is below a threshold, we claim our theoretical model is applicable now. This threshold is defined by simply computing the maximum and minimum loss with respect to the interpolating predictor, and taking the value 20% higher than minimum on the scale defined from minimum to maximum loss.

Fitting the Bayesian Model. We must perform the following three steps in order to fit our model.

- **Approximating Kolmogorov complexity.** Because true Kolmogorov complexity is not computable, we estimate an upper bound by compressing a self-contained bundle for each predictor: (i) the cleaned Python source that instantiates the predictor, and (ii) any numpy arrays it needs at inference time (e.g., the full table of urn distributions for the memorizing baseline in Balls & Urns). We remove comments, docstrings, and extraneous whitespace from the source code. For arrays, we first apply simple delta-encoding (store successive differences) to expose additional structure. The pre-processed bundle is compressed with four strong, off-the-shelf algorithms: `lzma` (`preset=9 | PRESET`$_{\texttt{EXTREME}}$), `bzip2` (`level=9`), `brotli` (`quality=11, mode=TEXT`), and `zstd` (`level=22`). We take the smallest compressed size (in bits) across the four algorithms as our estimate; this is a standard practice for obtaining a loose but practical upper bound. To keep estimates comparable, we exclude external libraries such as PyTorch from compression: all predictors call the same set of PyTorch primitives, so including them would add a large constant offset without altering relative complexities. This choice does, however, ignore the fact that some primitives might be cognitively "cheaper" for a Transformer to implement than others—an important caveat for future work.
- **Computation of average log likelihood per predictor.** For every experimental condition we first record the token-level log-likelihood that each predictor assigns to the in-distribution sequences, following the procedure described above. Because the irreducible error term cancels when models are compared, we restrict KL-based evaluations to the Balls & Urns and classification tasks, treating the linear-regression setting separately. To summarize performance we need the mean log-likelihood per token, yet the empirical loss distribution is, at times, quite skewed: tokens typically incur near-zero loss, whereas a small fraction of early tokens produce large spikes. A naive arithmetic mean therefore converges slowly and exhibits high variance. Thus, we instead

use median of means, an estimator for the true mean that has better convergence under long tailed distributions.

- **How fitting is done.** To fit the 3 free parameters of the Bayesian model, we minimize the mean KL divergence (or mean-squared error in the linear-regression setting) between the interpolated predictions and the Transformer outputs. Optimization is performed with `scipy.optimize.minimize` using the L-BFGS-B algorithm, capped at 1K iterations and 2K function evaluations, with gradient and function tolerances of 10^{-7}. Exact gradients are supplied via PyTorch's automatic differentiation, ensuring stable convergence. For each task we fit on $80\,\%$ of the (N, D) configuration grid and reserve the remaining $20\,\%$ for held-out validation and diagnostic checks.

G Additional Details Regarding Settings and Predictors

We now give a more detailed discussion of the different settings analyzed in this work: (i) Balls & Urns, (ii) Linear Regression, and (iii) Classification. We also provide details of how the memorizing and generalizing predictors are implemented for these settings. Broadly, as also visualized in Fig. 10, all settings involve learning of a mixture of tasks $\mathcal{T}_{\text{train}}$ drawn from the true task distribution $\mathcal{T}_{\text{true}}$. The number of tasks involved in the mixture is called its task diversity (denoted D). For all settings, we find models learn predictors of two types: a *memorizing predictor*, which corresponds to the Bayesian posterior predictive distribution with a discrete prior over seen tasks $\mathcal{T}_{\text{train}}$, and a *generalizing predictor*, which corresponds to the Bayesian posterior predictive distribution with a prior over the true task distribution $\mathcal{T}_{\text{true}}$. The precise forms of these predictors, as well as how sequences are assembled into training batches in each setting, are provided in the following sections.

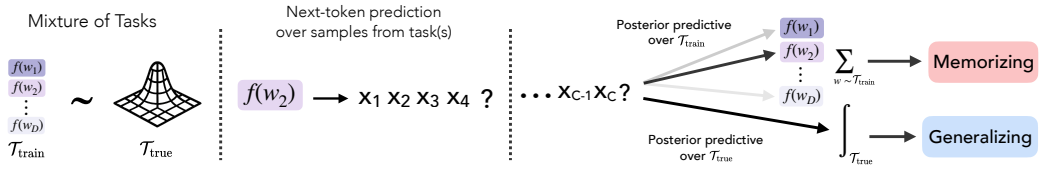

Figure 10: **General Abstraction Capturing our Experimental Settings and their Predictors.** Each setting involves a mixture of parameterized functions (called a "task"), with D functions (the "task diversity"). Task consist of predicting the next element in a sequence, and vary based on whether models are trained in a standard auto-regressive fashion (like Balls & Urns) or whether they are only trained to predict some elements in the sequence (only function outputs in Linear Regression, and only the last label in Classification). Across settings, the solutions learned by Transformers can be characterized as *memorizing predictors* or *generalizing predictors*. A memorizing predictor is defined as the Bayesian posterior predictive distribution with $\mathcal{T}_{\text{train}}$, the distribution of seen tasks, as its prior. A generalizing predictor is defined as the Bayesian posterior predictive with the true task distribution $\mathcal{T}_{\text{true}}$ as its prior.

G.1 Balls and urns

Memorizing Predictor. The memorizing predictor perform a Bayesian averaging operation and requires computing a weighted average of all urn distributions seen during training. The weight on each urn is derived from the likelihood of the current sequence of observations being generated by that urn. Formally, let $\boldsymbol{w}_d$ denote the parameters for an urn $d \in \{1, \ldots, D\}$, with each element $\boldsymbol{w}_d^{(k)}$ containing the probability for ball type $k \in \{1, \ldots, m\}$ under urn D. Then, the probability of a new ball being of type k after seeing a sequence $\boldsymbol{s}$ is:

$$p(k|\boldsymbol{s}) \propto \sum_{\boldsymbol{w}_d \in \mathcal{T}_{\text{train}}} \boldsymbol{w}_d^{(k)} \prod_{k' \in \{1,\ldots,m\}} (\boldsymbol{w}_d^{(k')})^{n_{k'}}.$$

With $n_{k'}$ being the number of occurrences of ball of type k' in the sequence.

Generalizing Predictor. Given that the true distribution is a uniform Dirichlet, and that the Dirichlet distribution is a conjugate prior of the categorical distribution (from which we draw our samples), the

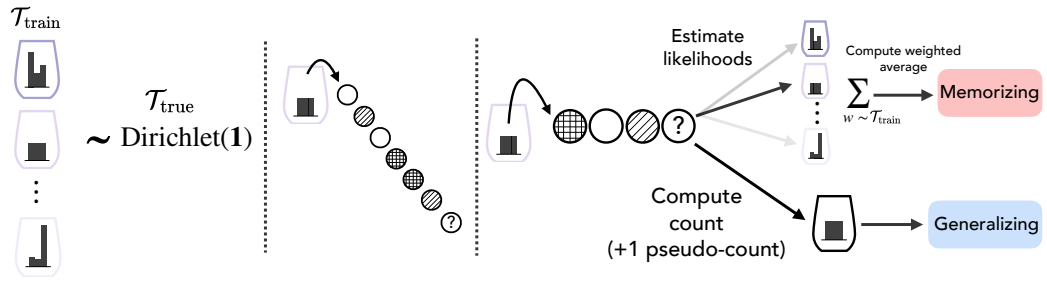

Figure 11: **Visualizing the setup for Balls and Urns.** Each task involves an "urn" that outputs a "ball" of a specific type every time it is sampled from. The task then involves seeing samples from an urn, concatenated to form a sequence. A memorizing predictor for this setting involves computing the sequence-level unigram statistics, i.e., the counts for each ball type, and comparing them with distributions from urns seen during training. Meanwhile, a generalizing predictor simply assumes the distribution of balls follows a uniform Dirichlet prior, thus predicting simply based on computing the unigram statistics from the sequence and adding a 1 pseudo-count for each ball type. Thus, this predictor generalizes to novel urns not seen by the model during training.

optimal way to estimate the probability of a ball of a particular type k in a sequence $\boldsymbol{s}$ of length C is: $p(k|\boldsymbol{s}) = \frac{n_k+1}{C}$, with n_k being the number of occurrences of ball of type k in the sequence. That is, the optimal strategy for the true distribution is simply computing a count for each type, adding a 1 pseudo-count, and dividing by the sequence length.

G.2 Linear regression

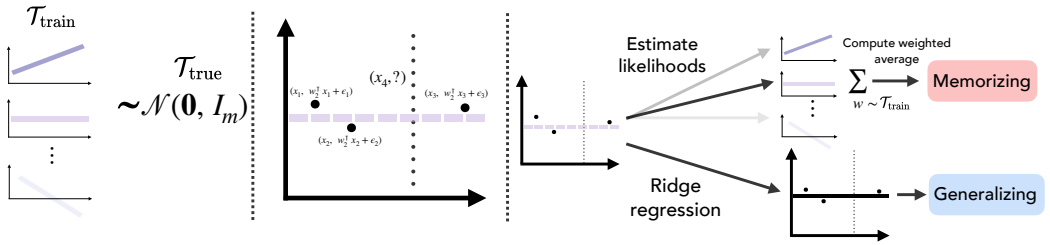

Figure 12: **Visualizing the setup for Linear Regression.** Each task involves a linear regression problem, defined by parameters $\boldsymbol{w}$, that outputs a pair $(\boldsymbol{x}, \mathrm{y})$, where $\mathrm{y} = \boldsymbol{w}^\intercal \boldsymbol{x} + \epsilon$ is a noisy linear transformation of the vector $\boldsymbol{x}$. The task then involves seeing a sequence of such pairs, concatenated to form a sequence. A memorizing predictor for this setting involves computing the likelihood of the pairs seen in context under the parameters of each task seen during training, using this result to compute a posterior over said tasks and a posterior-weighted average with a discrete prior over seen tasks. Meanwhile, the generalizing predictor is merely the ridge regression operation, which is equivalent to performing a Bayesian average operation assuming a continuous Gaussian prior. Correspondingly, this predictor generalizes to novel regression tasks that were not seen by the model during training.

Additional Details Not Provided in Main Text. To maintain a constant signal-to-noise ratio across tasks with different dimensionality m, we set $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, with $\sigma^2 = \frac{m}{256}$.

Generalizing Predictor. The *generalizing predictor* in this case simply performs ridge regression. Given $\boldsymbol{x} = (\boldsymbol{x}_1^\intercal, ..., \boldsymbol{x}_{C-1}^\intercal)$ and $\mathrm{y} = (\mathrm{y}_1, ..., \mathrm{y}_{C-1})$, the weight estimate is after seeing $C-1$ examples is:

$$\hat{\boldsymbol{w}}_G^{(C)} = (\boldsymbol{x}^\intercal \boldsymbol{x} + \sigma^2 \boldsymbol{I}_m)^{-1} \boldsymbol{x}^\intercal \mathrm{y}$$

Figure 13: **Visualizing the setup for Classification.** Each task involves noisy item-label pairs $\tilde{\boldsymbol{w}} \oplus \boldsymbol{l}$, and ends with a noisy query item $\tilde{\boldsymbol{w}}_{\text{query}}$ which comes from the same true item $\boldsymbol{w}$ as one of the items in the sequence. Items are noised via $\tilde{\boldsymbol{w}} = \frac{\boldsymbol{w}+\sigma\epsilon}{\sqrt{1+\sigma^2}}$, with $\epsilon \in \mathcal{N}(0, \boldsymbol{I}_m/m)$ sampled from the same distribution as the true item $\boldsymbol{w}$. A memorizing predictor knows the true items in the training distribution, computes the likelihood that the noisy query item comes from each true item, and accordingly computes a posterior-weighted average using the labels for each true item. Note that this predictor completely ignores the context, and hence was described as an 'in-weights learning' solution in previous works [24, 28]. In contrast, the generalizing predictor implements a noisy copy operation. It estimates the likelihood that the query head and each item seen in context come from the same true item. Then, it predicts via a posterior-weighted average according to the labels of each item seen in context. Therefore, this predictor works for OOD settings containing novel items that were not previously seen during training.

Memorizing Predictor. The *memorizing predictor* in this case performs inference by Bayesian averaging: a weighted average across all $\boldsymbol{w}^{(d)}$s seen in the training distribution, with weights determined by the likelihood that the sequence was generated by the specific $\boldsymbol{w}^{(t)}$. After seeing $C-1$ examples, the weight estimate is:

$$\hat{\boldsymbol{w}}_M^{(C)} = \sum_{\boldsymbol{w}_d \in \mathcal{T}_{\text{train}}} \frac{\exp(-\frac{1}{2\sigma^2}\sum_{c=1}^{C-1}(y_c - \boldsymbol{w}_d^\intercal\, \boldsymbol{x}_c)^2)}{\sum_{\boldsymbol{w}_{d'} \in \mathcal{T}_{\text{train}}} \exp(-\frac{1}{2\sigma^2}\sum_{c=1}^{C-1}(y_c - \boldsymbol{w}_{d'}^\intercal\, \boldsymbol{x}_c)^2)} \boldsymbol{w}_d$$

G.3 Classification

We use the classification setting with the formulation from [29] as well as inspiration from the noisy class centroids introduced by [28]. As Nguyen and Reddy [29] have shown, their simplified setting captures the phenomenology of other classification settings proposed by Chan et al. [20], Reddy [28]. For simplicity, we include only binary labels in our version.

Additional Details Not Provided in Main Text. When presented in context, items $\boldsymbol{w}$ are noised and presented as $\tilde{\boldsymbol{w}} = \frac{\boldsymbol{w}+\sigma\epsilon}{\sqrt{1+\sigma^2}}$. We use within-class variance of $\sigma^2 = 0.5$ in all settings, and $\epsilon \in \mathcal{N}(0, \boldsymbol{I}_m/m)$ is sampled separately for each item in the context.

Memorizing Predictor. The *memorizing predictor* in this setting performs inference by computing a posterior-weighted average over item-label pairs seen in the training distribution $\mathcal{T}_{\text{train}}$, i.e., $\boldsymbol{w}_1 \oplus \boldsymbol{l}_1, \ldots, \boldsymbol{w}_D \oplus \boldsymbol{l}_D$. The form for a noisy item used for defining the input sequence is $\tilde{\boldsymbol{w}} = \frac{\boldsymbol{w}+\sigma\epsilon}{\sqrt{1+\sigma^2}}$ for some $\boldsymbol{w} \sim \mathcal{T}_{\text{train}}$. Thus, since ϵ has covariance $\boldsymbol{I}_m/m$, we can write a noisy item sampled from a given $\boldsymbol{w}$ will be distributed as: $(\tilde{\boldsymbol{w}} \mid \boldsymbol{w} = \boldsymbol{w}_d) \sim \mathcal{N}(\frac{1}{\sqrt{1+\sigma^2}}\boldsymbol{w}, \frac{\sigma^2}{1+\sigma^2}\ \boldsymbol{I}_m/m)$. The probability of the query label being 1 can then be defined as follows:

$$\begin{aligned} p(1|\boldsymbol{s}) &\propto \sum_{\boldsymbol{w}_d \in \mathcal{T}_{\text{train}}} p(\tilde{\boldsymbol{w}}_{\text{query}}|\boldsymbol{w}_d)\ p(1|\boldsymbol{w}_d) \\ &\propto \sum_{\boldsymbol{w}_d \in \mathcal{T}_{\text{train}}} \exp\left(-\frac{m}{2\sigma^2}\ (1+\sigma^2)\ \left\|\tilde{\boldsymbol{w}}_{\text{query}} - \frac{1}{\sqrt{1+\sigma^2}}\boldsymbol{w}_d\right\|^2\right)\ \mathbb{1}(\boldsymbol{l}_d = 1), \end{aligned}$$

where we disregard constants outside the exp term.

Generalizing Predictor. The *generalizing predictor* in this setting performs inference by computing a posterior-weighted average over item-label pairs in the context. Specifically, note that we define sequences using noised versions $\tilde{\boldsymbol{w}}$ of task vectors $\boldsymbol{w} \sim \mathcal{T}_{\text{true}}$. Importantly, we allow sampling with replacement, i.e., the same task $\boldsymbol{w}$ can be used to define multiple item-label pairs. This can be thought of as a biased sampling process, instead of the random sampling one, whereby a task seen in-context has finite odds of being seen again in the sequence than a random one. Accordingly, the prior will collapse onto just the seen item pairs (as it will be infinitesimally small for items sampled randomly). We thus merely need to compute the joint probability of the query item given the items seen in context, leading to the following form:

$$p(1|\boldsymbol{s}) \propto \sum_{\tilde{\boldsymbol{w}}_d \in \text{Context}} \exp\left(-\frac{m\ (1+\sigma^2)^2}{2\sigma^2\ (2+\sigma^2)} \left\| \tilde{\boldsymbol{w}}_{\text{query}} - \frac{\tilde{\boldsymbol{w}}_d}{1+\sigma^2} \right\|^2 \right)\ \mathbb{1}(\boldsymbol{l}_d = 1).$$

H Main Results Across All Settings

In the following sections, we provide the results reported in the main paper across all settings and experiments.

H.1 Task Diversity Threshold

We find the phenomenon of a task diversity threshold [25] to be very robust across settings and experimental conditions. More specifically, we consistently find that increasing task diversity yields a transition in Transformer behavior from behaving like a memorizing predictor to behaving like a generalizing predictor. In the following, we present evidence of this phenomenon for ID sequences. See results in following pages.

H.1.1 Balls & Urns

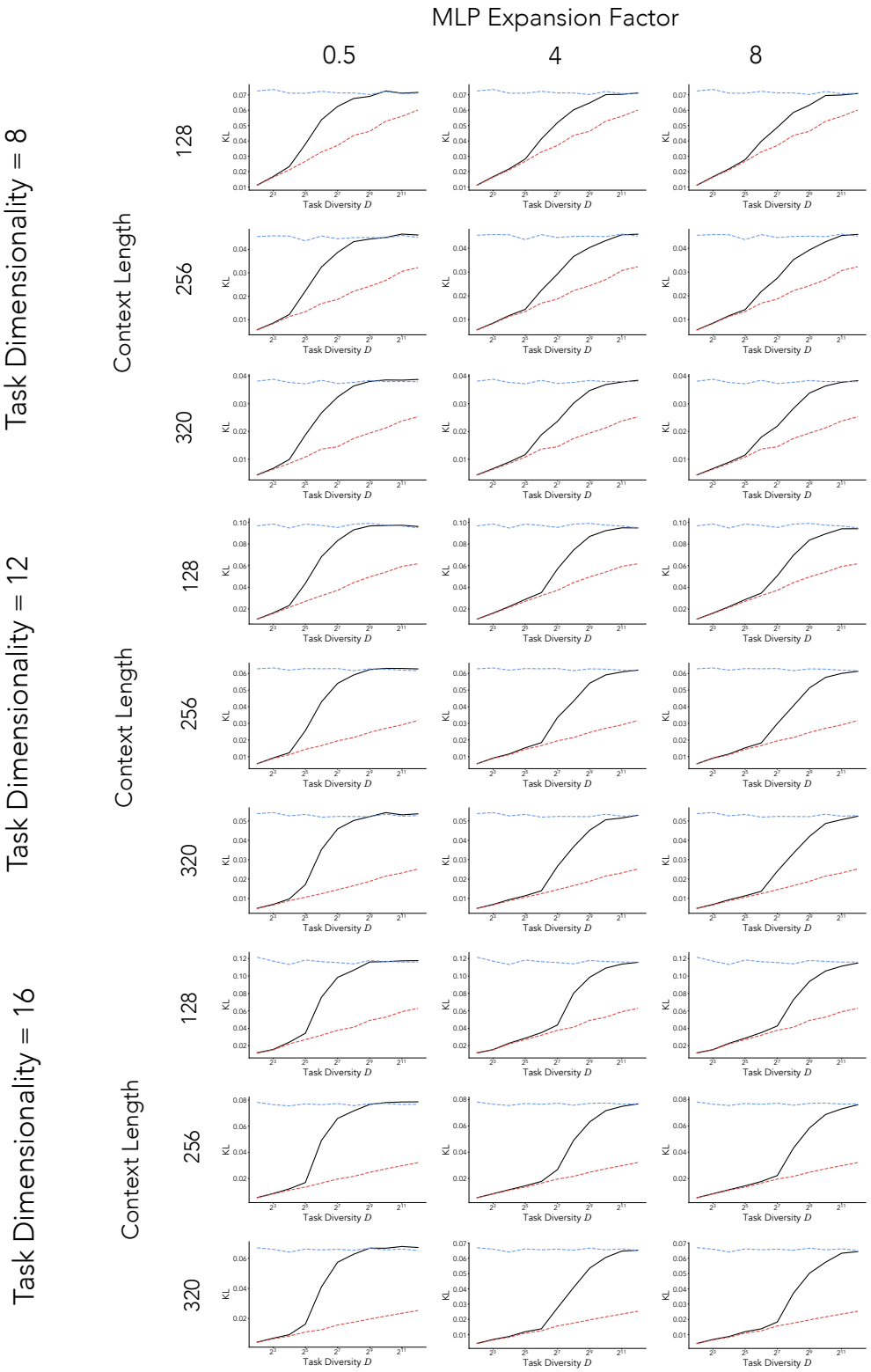

Figure 14: **Task Diversity Threshold across Balls & Urns conditions.** Red dashed line indicates the memorizing solution M, blue dashed line indicates the generalizing solution G, and black solid line indicates Transformer behavior at the end of training (100K steps).

H.1.2 Linear Regression

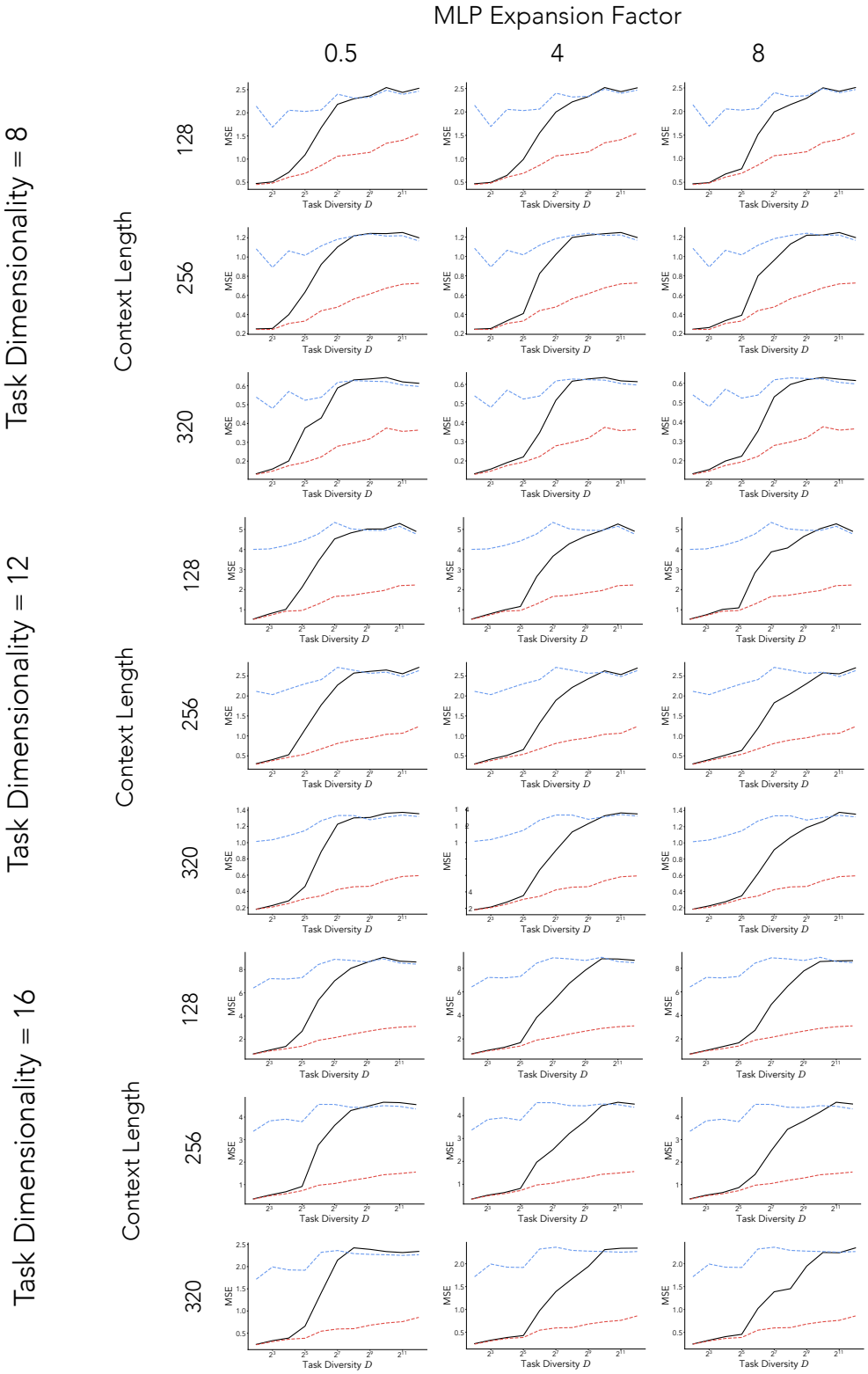

Figure 15: **Task Diversity Threshold across Linear Regression conditions.** Red dashed line indicates the memorizing solution M, blue dashed line indicates the generalizing solution G, and black solid line indicates Transformer behavior at the end of training (100K steps).

H.1.3 Classification

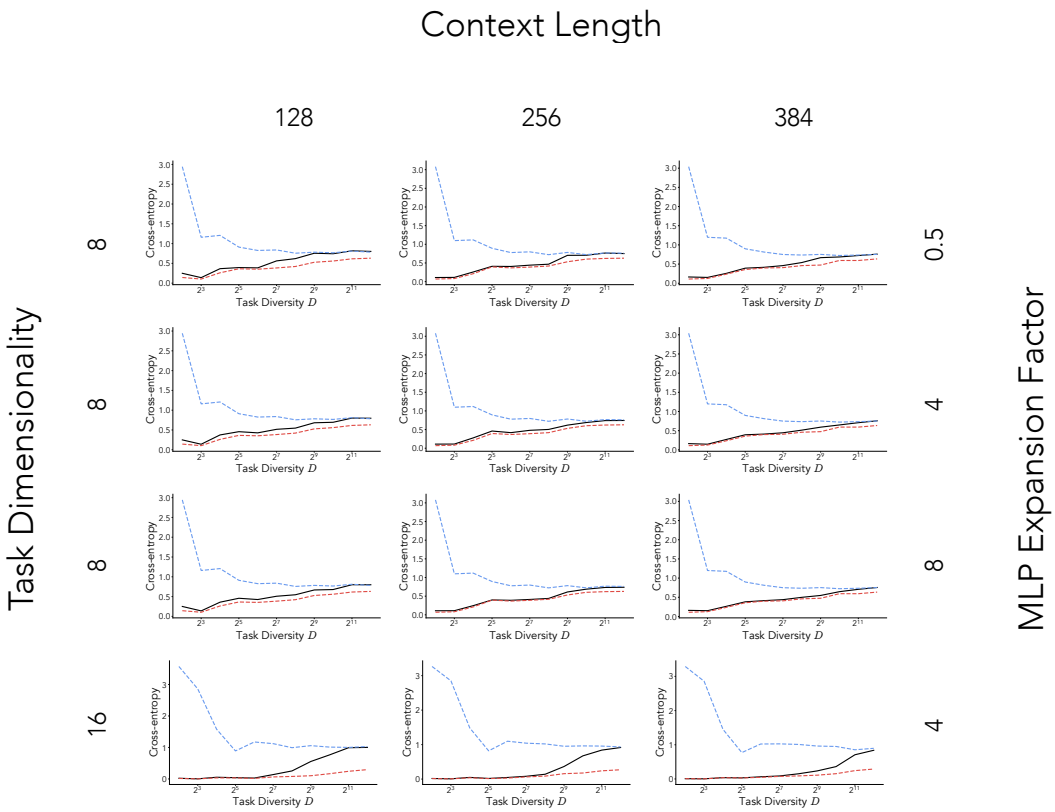

Figure 16: **Task Diversity Threshold across Classification conditions.** Red dashed line indicates the memorizing solution M, blue dashed line indicates the generalizing solution G, and black solid line indicates Transformer behavior at the end of training (100K steps). IWL evaluation presented.

H.2 Transience

Across settings and conditions, we also find the phenomenon of transience [24] to be consistent in moderate task diversity values. More specifically, in moderate task diversity values, we see the Transformer approach the generalizing solution in terms of OOD performance early in training, only to eventually begin memorizing and worsen in OOD performance. In the figures below, we show OOD performance of Transformers trained in different task diversity conditions, with low task diversity values showing immediate memorization, moderate task diversity values showing transience, and high task diversity values often continuing to generalize well throughout training. See results in following pages.

H.2.1 Balls & Urns

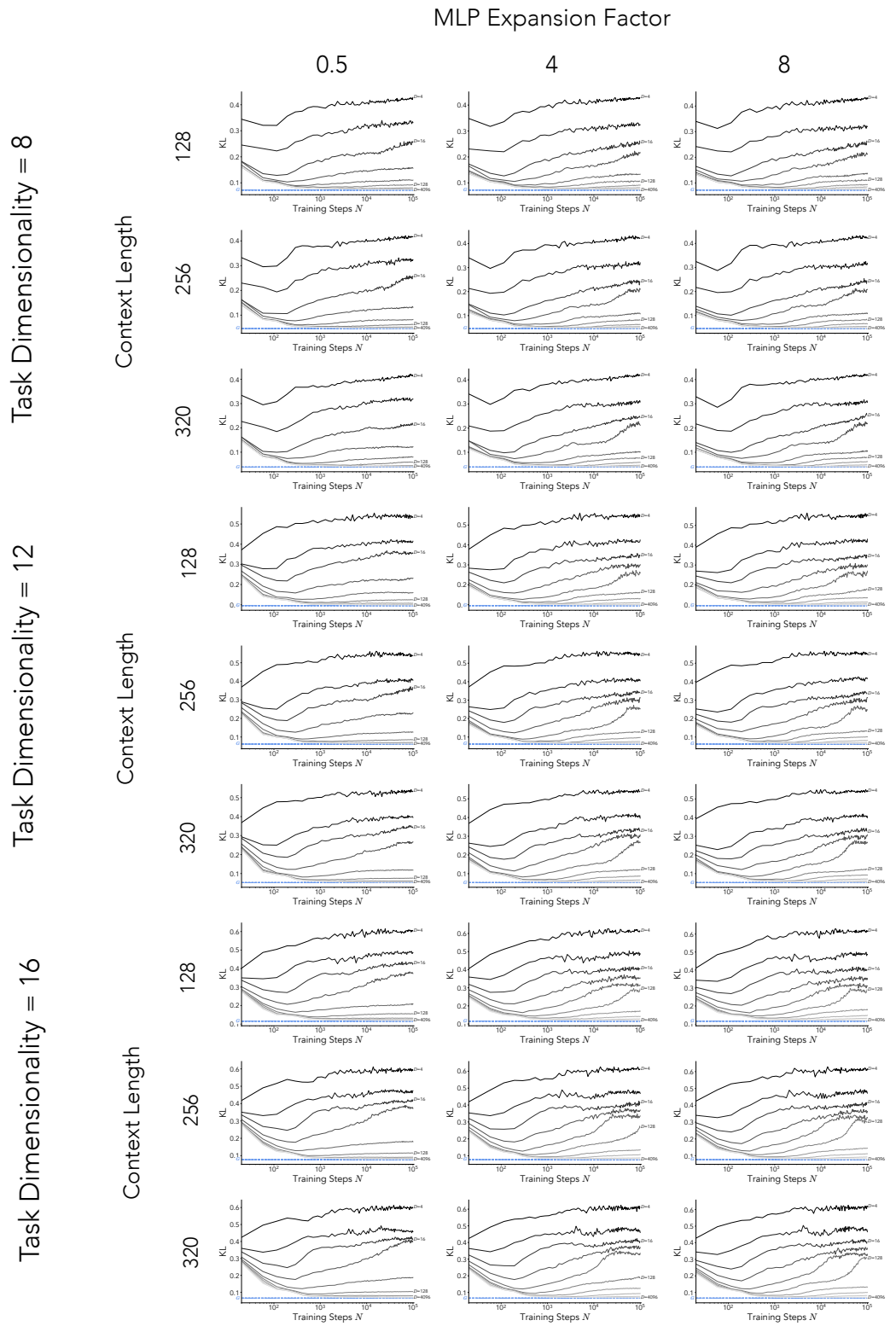

Figure 17: **Transience across Balls & Urns conditions.** OOD performance presented. Blue Dashed line indicates OOD performance of generalizing solution G.

H.2.2 Linear Regression

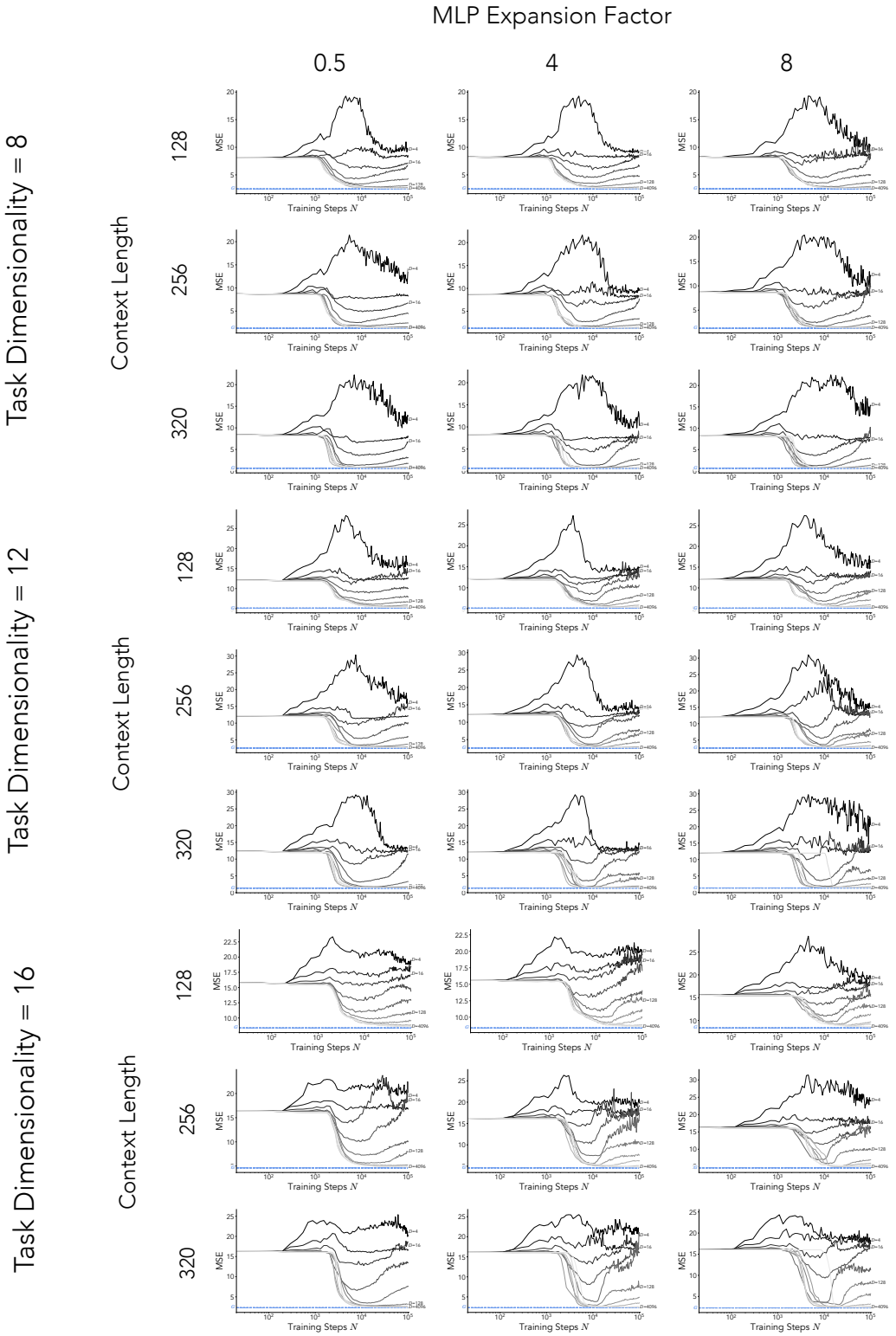

Figure 18: **Transience across Linear Regression conditions.** OOD performance presented. Blue Dashed line indicates OOD performance of generalizing solution G.

H.2.3 Classification

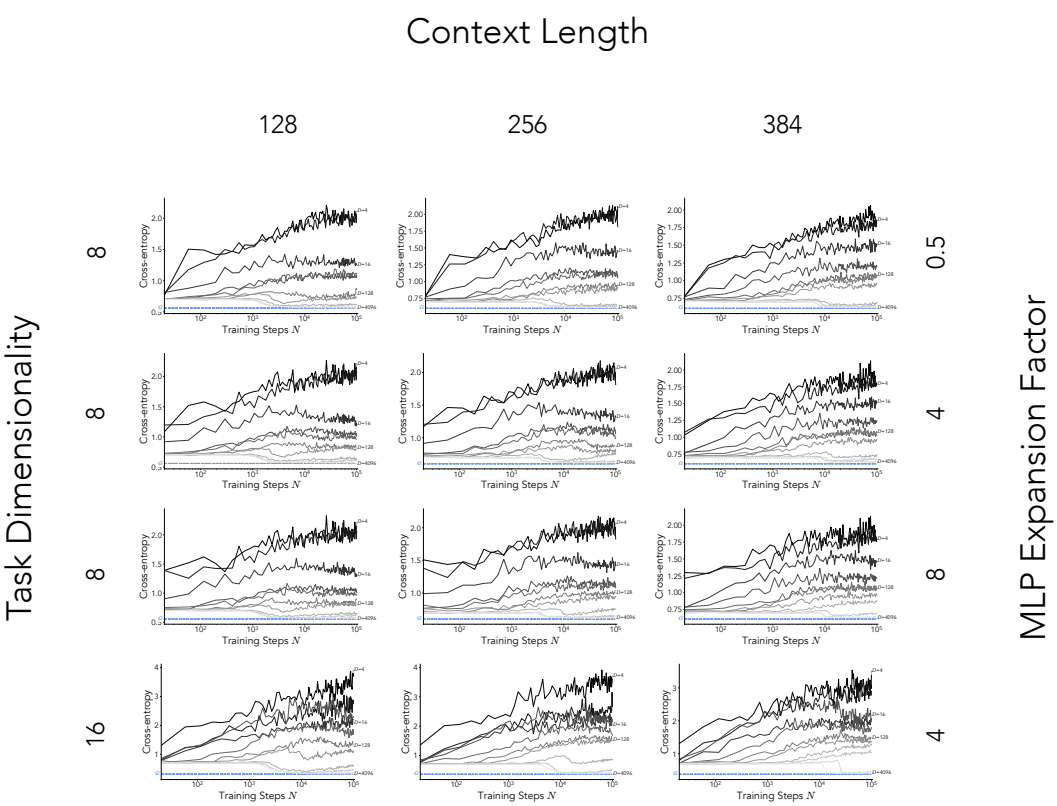

Figure 19: **Transience across Classification conditions with 8 dimensions.** OOD performance presented. Blue Dashed line indicates OOD performance of generalizing solution G. Note that in the case of classification, it is often the case that only one or two task diversity conditions show transient generalization, as can be seen more clearly from the absolute distance maps in the next section.

H.3 Absolute Distance from Predictors

We find that across settings and conditions, Transformers primarily learn and transition between behaving like two predictors: a generalizing solution G, which consists of the Bayesian posterior predictive distribution over the true task distribution $\mathcal{T}_{\text{true}}$ and a memorizing solution M which consists of the Bayesian posterior predictive distribution over the training task distribution $\mathcal{T}_{\text{train}}$. In the figures below, we display the absolute distance from each of these predictors, as well the relative distance in the background (ranging from red indicating closeness to M to blue indicating closeness to G). See results in following pages.

H.3.1 Balls & Urns

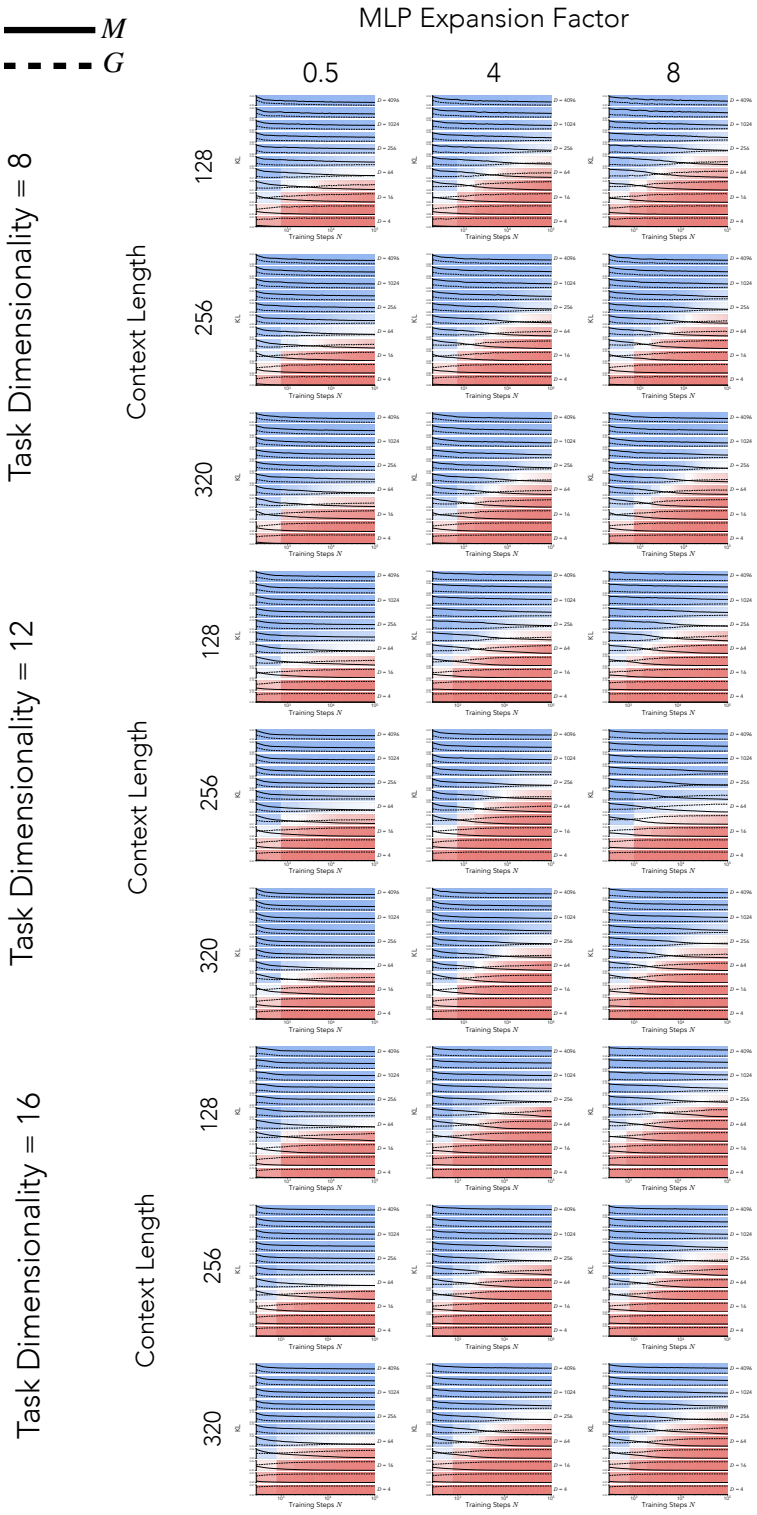

Figure 20: **Absolute and relative distance from predictors across Balls & Urns conditions.** Distance from the generalizing solution G shown in the dashed black line, while distance from the memorizing solution M is shown in the solid line. KL indicates symmetrized KL divergence (average of forward and backward KL).

H.3.2 Linear Regression

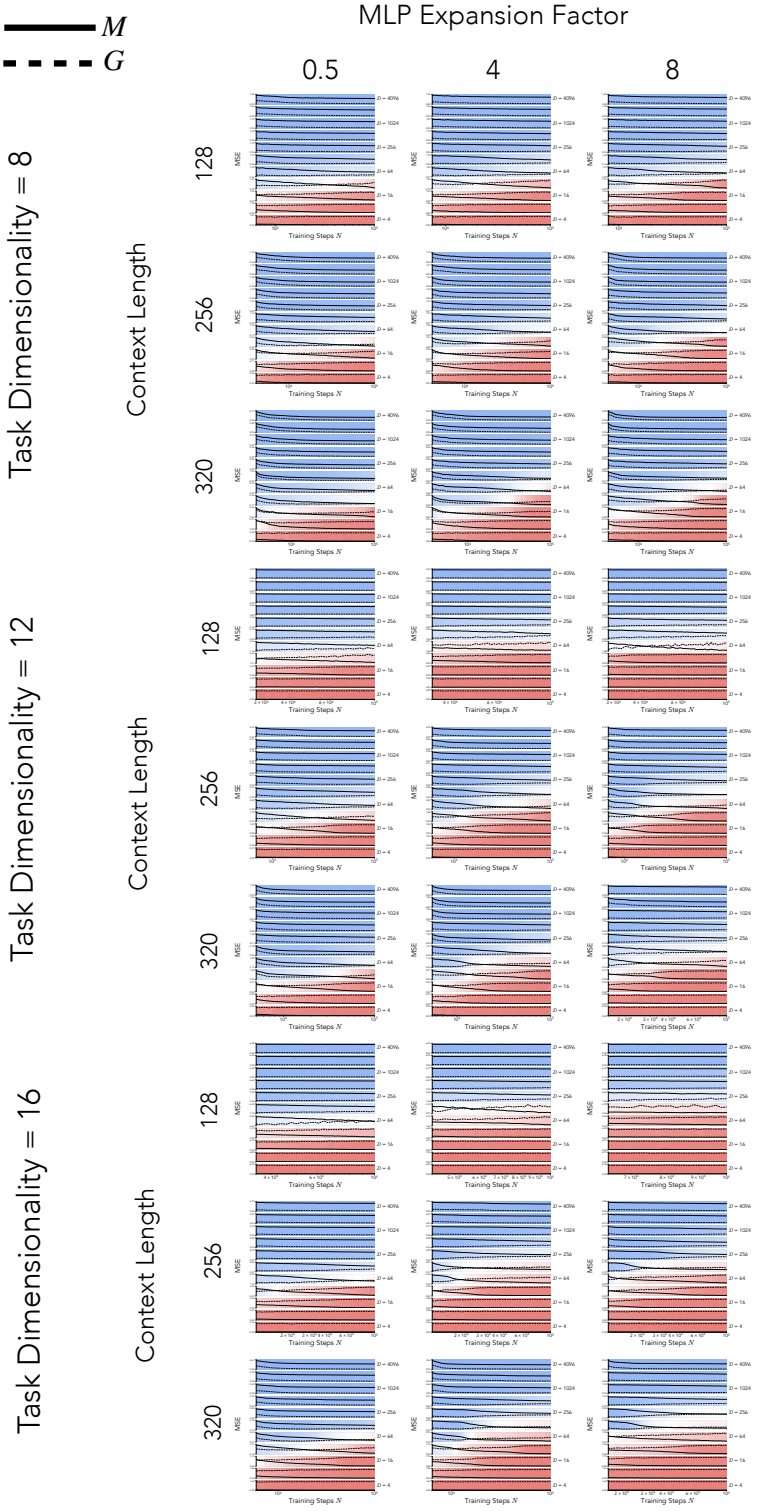

Figure 21: **Absolute and relative distance from predictors across Linear Regression conditions.** Distance from the generalizing solution G shown in the dashed black line, while distance from the memorizing solution M is shown in the solid line.

H.3.3 Classification

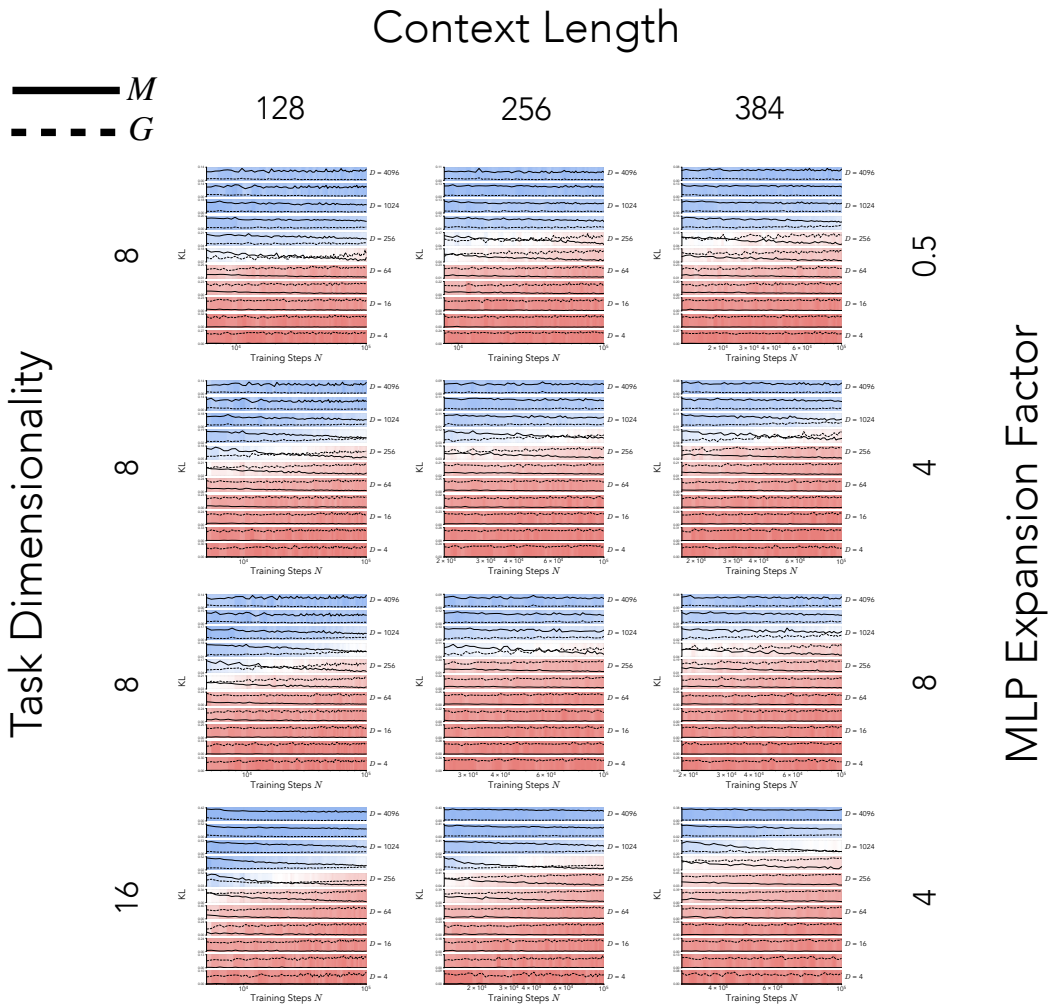

Figure 22: **Absolute and relative distance from predictors across Classification conditions.** Distance from the generalizing solution G shown in the dashed black line, while distance from the memorizing solution M is shown in the solid line. KL indicates symmetrized KL divergence (average of forward and backward KL).

H.4 Model Predictions

Across settings and training conditions, we find that our model's predictions consistently perform well both in estimating the next-token prediction behavior of the Transformer, as well as capturing its change in generalization behavior across conditions, as displayed by the relative distance from predictors G and M. We conduct thorough stress-testing of our model across 3 settings and 66 (N, D) maps, each with above 900 model checkpoints, and find that our model consistently performs well across nearly all maps, thus providing robust evidence for the predictive and explanatory power of our account. See results in following pages.

H.4.1 Balls & Urns

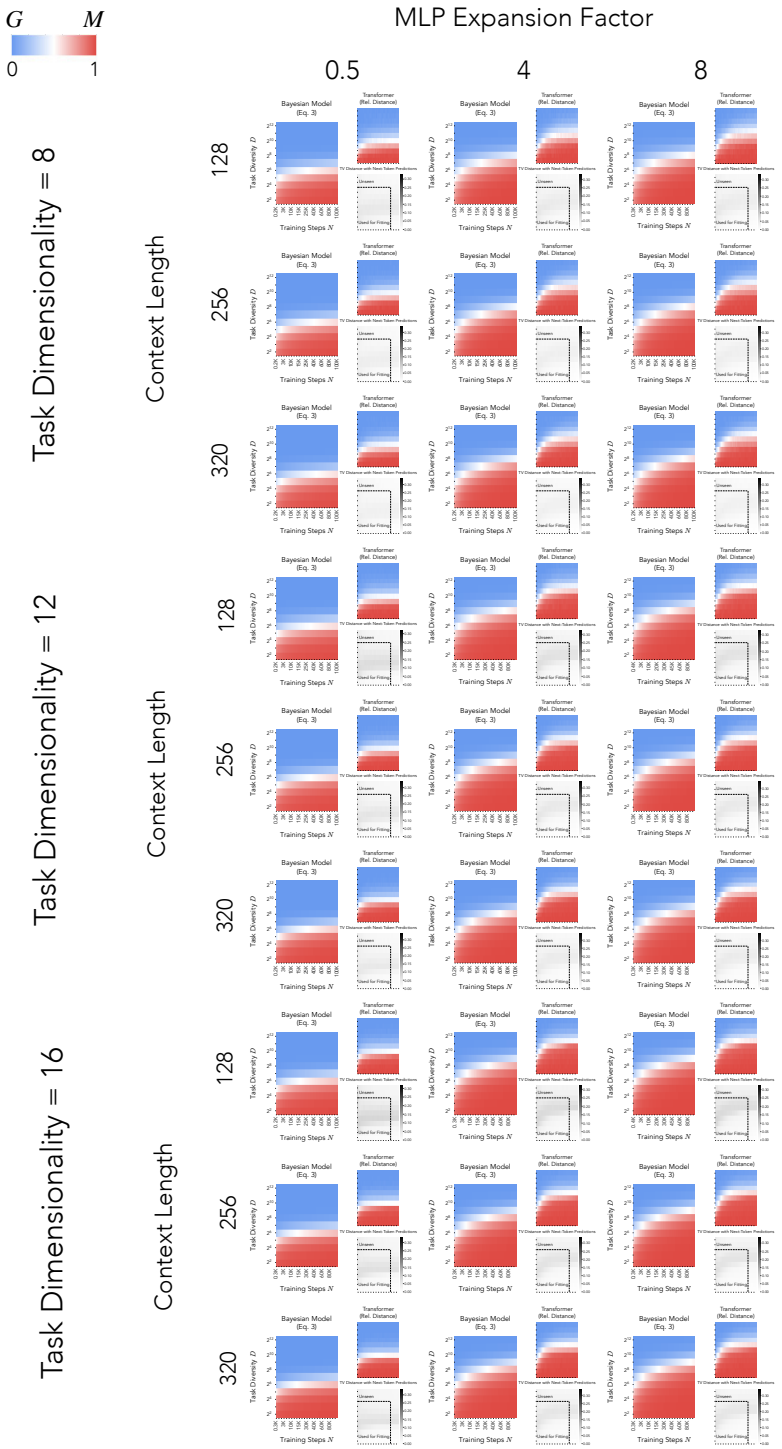

Figure 23: **Bayesian model predictions across Balls & Urns conditions.** Red indicates closeness to memorizing predictor M, while blue indicates closeness to generalizing predictor G. Shown is a comparison between the posterior probability of the memorizing solution M given by our Bayesian model (left) and the relative distance from the Transformer (top right), as well as heatmaps indicating similarity with Transformer next-token predictions (bottom right). Max color bar value is determined by the performance of a baseline predictor that always outputs the mean of the distribution $\mathcal{T}_{\text{true}}$.

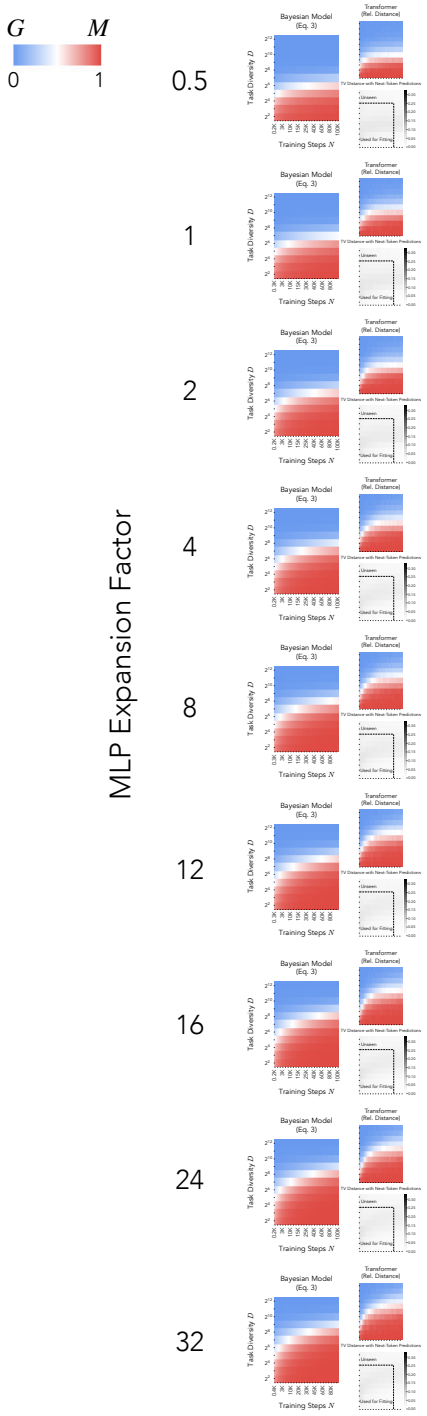

Figure 24: **Bayesian model predictions across Balls & Urns conditions with varying MLP expansion factors.** Red indicates closeness to memorizing predictor M, while blue indicates closeness to generalizing predictor G. Shown is a comparison between the posterior probability of the memorizing solution M given by our Bayesian model (left) and the relative distance from the Transformer (top right), as well as heatmaps indicating similarity with Transformer next-token predictions (bottom right). Context length is 128, task dimensionality is 8, and hidden size is 64 in all conditions shown. MLP width is given by hidden size times MLP expansion factor. Max color bar value is determined by the performance of a baseline predictor that always outputs the mean of the distribution $\mathcal{T}_{\text{true}}$.

H.4.2 Linear Regression

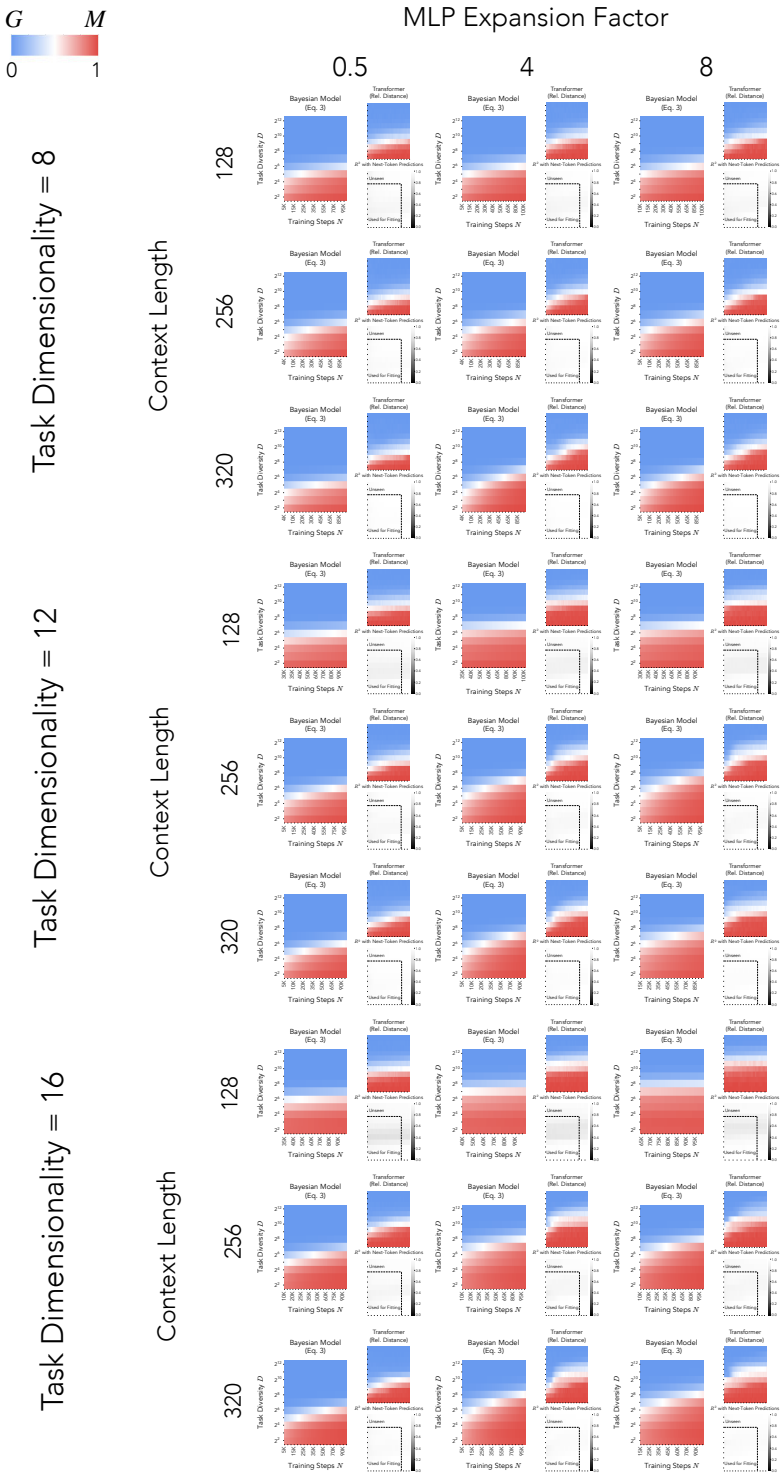

Figure 25: **Bayesian model predictions across Linear Regression conditions.** Red indicates closeness to memorizing predictor M, while blue indicates closeness to generalizing predictor G. Shown is a comparison between the posterior probability of the memorizing solution M given by our Bayesian model (left) and the relative distance from the Transformer (top right), as well as heatmaps indicating similarity with Transformer next-token predictions (bottom right).

H.4.3 Classification

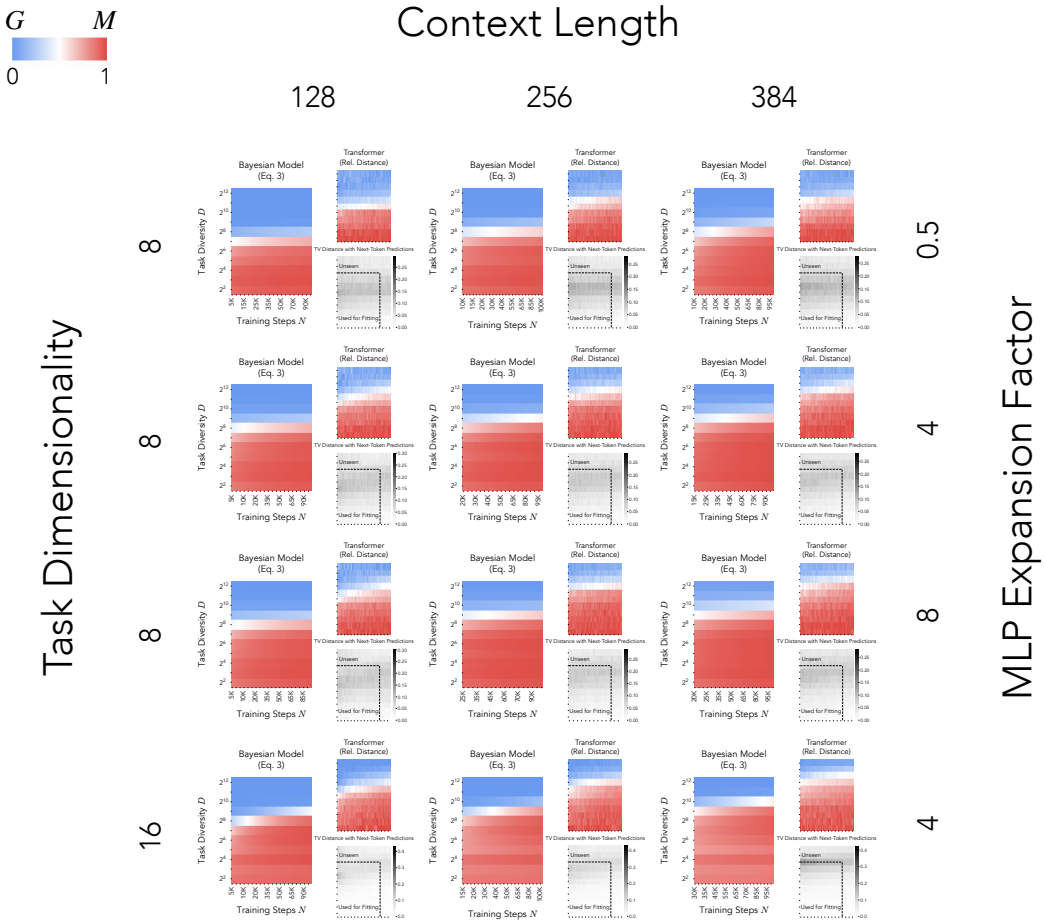

Figure 26: **Bayesian model predictions across Classification conditions.** Red indicates closeness to memorizing predictor M, while blue indicates closeness to generalizing predictor G. Shown is a comparison between the posterior probability of the memorizing solution M given by our Bayesian model (left) and the relative distance from the Transformer (top right), as well as heatmaps indicating similarity with Transformer next-token predictions (bottom right). Max color bar value is determined by the performance of a baseline predictor that always outputs the mean of the distribution $\mathcal{T}_{\text{true}}$. The conditions where task dimensionality equals 16 in this setting (bottom row) reveal a limitation of our complexity measure: since the memorizing and generalizing predictors are very close in performance in low task diversities for these conditions, the loss term does not strongly bias the Transformer towards the memorizing predictor. However, the Transformer, is very close to the memorizing predictor for low task diversities, which would indicate according to our framework that memorizing few items is substantially simpler than implementing a copy operation. However, this is not captured by our complexity measure, since the compressed size of the code for the memorizing and generalizing predictors is roughly the same, thus we are unable to capture the bias toward the memorizing predictor in low task diversity settings. To overcome this, in these 3 conditions only, we heuristically multiply the bit size of the code for the generalizing predictor by 5, and with that fix, we find good performance (though as can be seen, the model still under-weights the memorizing solution for some low task diversity conditions).

I Functional Form Ablations

Our functional form for the log posterior odds η consists of 3 free parameters, α, determining sublinear sample efficiency, β, a power law on the estimated Kolmogorov complexity K, and γ, a coefficient for the loss term. We find that each of these free parameters are necessary for the success of our model, since removing any of them results in a worsening of the model's ability to capture the phenomenology of ICL.

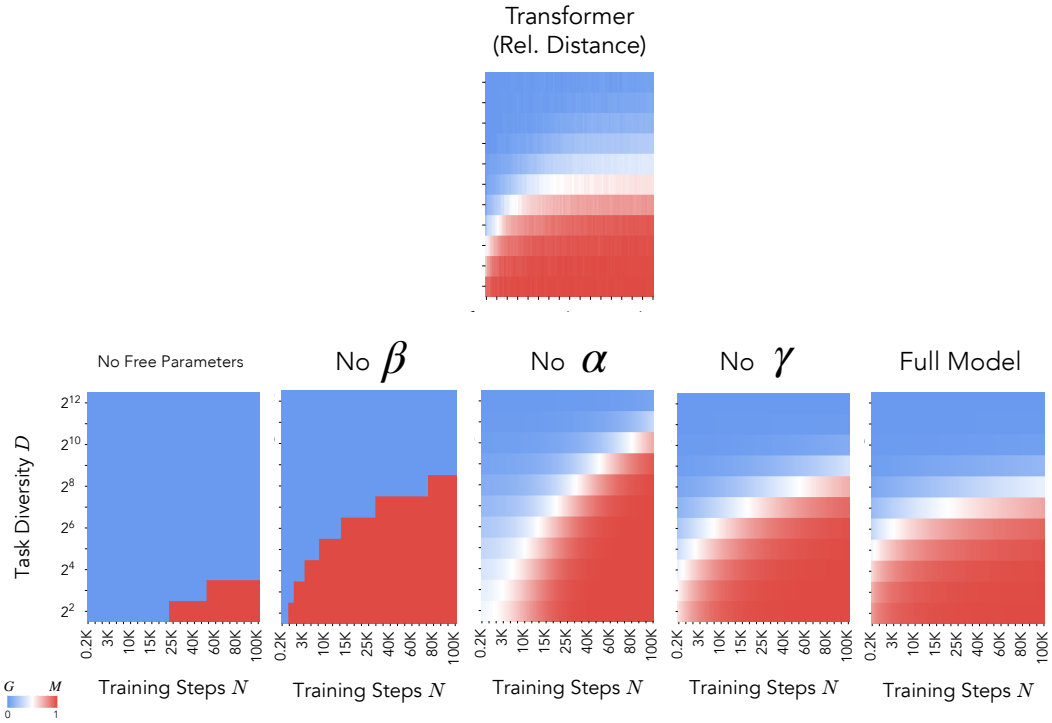

Figure 27: **Power laws over simplicity bias and sample efficiency are necessarily for explaining ICL phenomenology.** By ablating our functional form, we see that the free parameters α, β, γ are required for the performance of the model. In particular, the simplicity bias derived from K without a β term is much too sharp and over-penalizes complexity compared to the Transformer. Additionally, memorization proceeds much too rapidly without the α term, pointing toward the necessity of assuming sub-linear sample efficiency for capturing Transformer training dynamics.

J Memorization Continues to Increase After Task Diversity Threshold - Refutation of Raventós et al. [25]'s Claim

In Fig. 28, we show a refutation of the claim made by Raventós et al. [25], who claimed that after the task diversity threshold is reached, the Transformer will only continue to get closer to the generalizing solution throughout training, *regardless* of how long one trains. Our relative distance measure shows this to be false (see right side of Fig. 28). Even in conditions in which the task diversity threshold was reached and generalization is sustained throughout a reasonable amount of training (100K steps), we see that by absolute distance, the Transformer not only gets closer to the generalizing solution during training, it *also* gets closer to the memorizing solution during training (left side of Fig. 28). It seems that the rate at which the Transformer nears the memorizing solution is greater than that at which it nears the generalizing solution in task diversity settings such as $D = 512$ or $D = 256$, which is why the relative distance continues to grow even though the transformer continues to get closer to the generalizing solution. Note also, that in settings that clearly show transience, e.g., $D = 64$, the distance from the generalizing solution shrinks until a certain critical point in which it begins to grow. It is likely that this point can be reached in all task diversity settings examined given the growth trend of the relative distance. However, it may require a very long training process to reach that point.

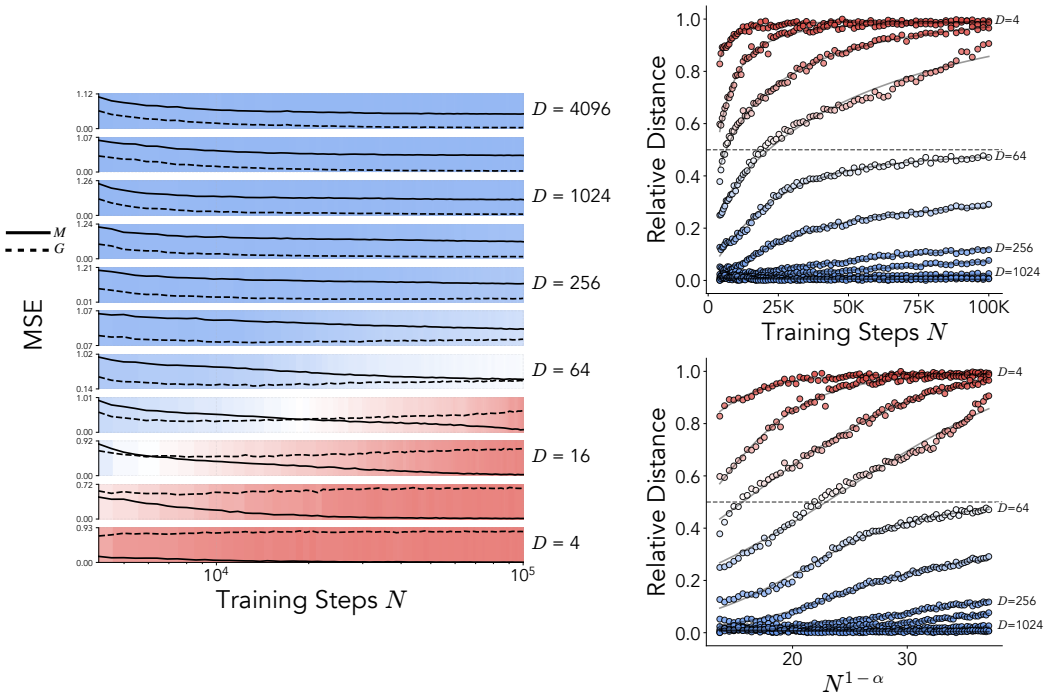

Figure 28: **Relative distance continues to rise throughout training, even after the task diversity threshold.** The figure displays relative distance, as well as absolute distance from the memorizing and generalizing solutions, for the linear regression setting with context length of 32, MLP expansion factor of 4, and 8 dimensions.

K Learning Rate Annealing Can Improve Adherence to Bayes-Optimal Trajectories

In our main experimental settings, in which we train with a constant learning rate, we find that relative distance trajectories follow a sigmoidal growth pattern with respect to $N^{1-\alpha}$ (see Fig. 6 and Fig. 29(a) top and bottom right). However, in contrast with our theory's predicted sigmoidal curves which plateau at 1 (i.e., *some* amount of training would yield full adherence to the memorizing solution, when it has a lower loss than the generalizing solution), this does not seem to be the case with the trajectories displayed by Transformers, which appear to plateau *early* (see Fig. 29(a) top right $D = 256$ for a clear example). Indeed, fitting parameterized logistic curves to these trajectories yields plateau values different from 1. To explain this, we turn to foundational work from Geman and Geman [87], who showed that a slow (logarithmic) temperature cooling schedule for Gibbs sampling substantially increases the likelihood of convergence to the global minimum (MAP estimate). Drawing a very rough parallel to the case of deep learning, it is reasonable to assume that a learning rate annealing schedule of some form is required to converge to the MAP estimate (which is the memorizing solution, in cases where it has lower loss than the generalizing solution). To test this, we repeated experiments in the Balls & Urns and Linear Regression settings and used warm-up and inverse-squared learning rate decay. Surprisingly, we indeed find that adding learning rate annealing can increase adherence to Bayes-optimal trajectories (Fig. 29(a)). However, we also find this effect is highly sensitive to training conditions: training for longer, even in conditions that yield the effect for a smaller number of training steps, can lead to plateau (Fig. 29(b), top), and slight changes in training conditions, in this case number of warm-up steps, can substantially reduce the effect (Fig. 29(b), bottom). It should also be noted that in this case, adherence to the Bayes-optimal trajectory is actually *negative* for generalization, since it means the model will converge to the memorizing solution quicker. However, this is not necessarily the case in more realistic training regimes, where the ability to overcome a simplicity bias and adopt more complex solutions is likely beneficial.

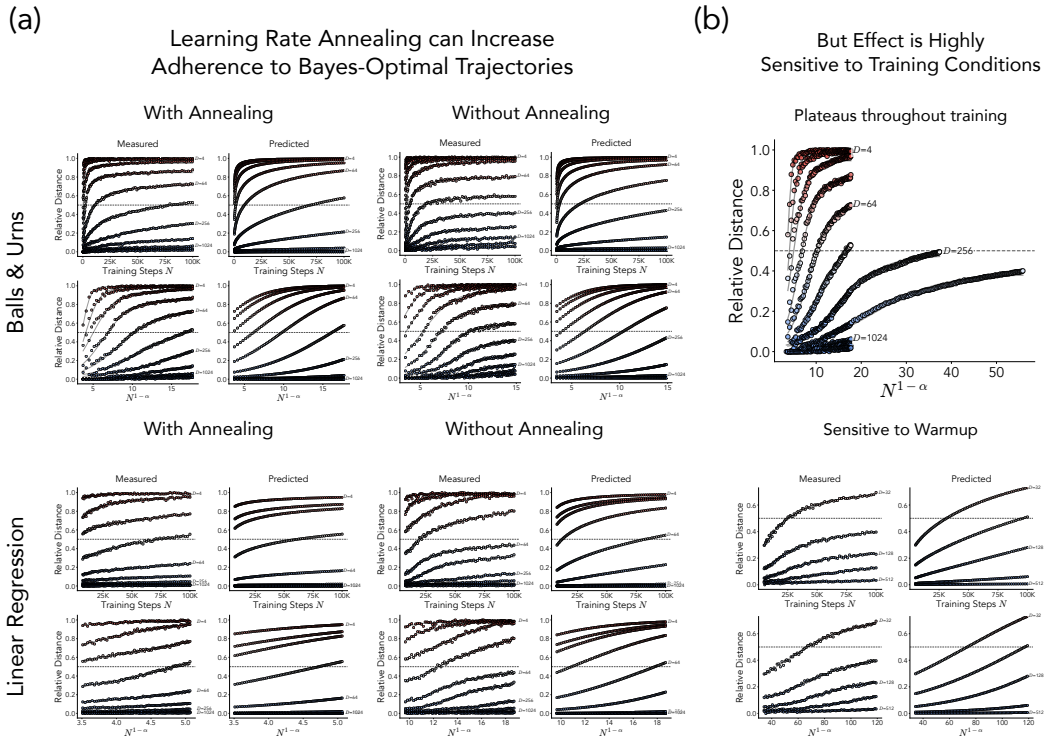

Figure 29: **Learning Rate Annealing Can Increase Adherence to Bayes-Optimal Trajectories.** (a) Balls & Urns setting uses context length of 128, MLP width of 256, and 8 dimensions. Linear regression uses similar variables except a context length of 16. (b) Effect is highly sensitive to training conditions: e.g., Training in the Linear regression setting with 5000 warmup steps failed, but succeeds with 500 warmup steps (the number used for the experiment in panel (a)).