Surprising Deviations from Bayesian View in In-Context Learning

Madhur Panwar
Microsoft Research India
`t-mpanwar@microsoft.com`

Kabir Ahuja
University of Washington
`kahuja@cs.washington.edu`

Navin Goyal
Microsoft Research India
`navingo@microsoft.com`

Abstract

In-context learning (ICL) is one of the surprising and useful features of large language models and subject of intense research. Recently, stylized meta-learning-like ICL setups have been devised that train transformers on sequences of input-output pairs $(x, f(x))$ using the language modeling loss. The function f comes from a function class and generalization is checked by evaluation on sequences for unseen functions from the same class. One of the main discoveries in this line of research has been that for several function classes, such as linear regression, transformers successfully generalize to new functions in the class. However, the inductive biases of these models resulting in this behavior are not clearly understood. A model with unlimited training data and compute is a Bayesian predictor: it learns the pretraining distribution. In this paper we empirically examine how far this Bayesian perspective can help us understand ICL. To this end, we generalize the previous meta-ICL setup to hierarchical meta-ICL setup which involve unions of multiple task families. We instantiate this setup on multiple function families and find that transformers can do ICL in this setting as well. We make some surprising observations: Transformers can learn to generalize to new function classes that were not seen during pretraining. This requires pretraining on a very small number of function classes and involves deviating from the Bayesian predictor on the pretraining distribution. Further, we discover the phenomenon of 'forgetting', where over the course of pretraining under hierarchical meta-ICL setup, the transformer first generalizes to the full distribution of tasks and later forgets it while fitting the pretraining distribution.

1 Introduction

In-context learning (ICL) is one of the major ingredients behind the astounding performance of large language models (LLMs) Brown et al. [2020], Touvron et al. [2023]. Unlike traditional supervised learning, ICL is the ability to learn new functions f without weight updates from input-output examples $(x, f(x))$ provided as input at the test time; in other words, learning happens *in context*. For instance, given the prompt `up -> down, low -> high, small ->`, a pretrained LLM will likely produce output `big`: it apparently infers that the function in the two examples is the antonym of the input and applies it on the new input. This behavior often extends to more sophisticated and novel functions unlikely to have been seen during training and has been the subject of intense study, e.g., Min et al. [2022b], Webson and Pavlick [2022], Min et al. [2022a], Liu et al. [2023], Dong et al. [2023]. More broadly than its applications in NLP, ICL can also be viewed as providing a method for meta-learning Hospedales et al. [2022] where the model learns to learn a class of functions.

Theoretical understanding of ICL is an active area of research. Since the real-world datasets used for LLM training are difficult to model theoretically and are very large, ICL has also been studied in stylized setups, e.g., Xie et al. [2022], Chan et al. [2022], Garg et al. [2022], Wang et al. [2023],

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Hahn and Goyal [2023]. These setups study different facets of ICL. In this paper, we focus on the meta-learning-like framework of Garg et al. [2022]. Unlike in NLP where training is done on documents for the next-token prediction task, here the training and test data look similar in the sense that the training data consists of input of the form $\big(\boldsymbol{x}_1, f(\boldsymbol{x}_1), \ldots, \boldsymbol{x}_k, f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1}\big)$ and output is $f(\boldsymbol{x}_{k+1})$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and are chosen i.i.d. from a distribution, and $f : \mathbb{R}^d \to \mathbb{R}$ is a function from a family of functions, for example, linear functions or shallow neural networks. We call this setup **MICL**. A striking discovery in Garg et al. [2022] was that for several function families, transformer-based language models during pretraining learn to implicitly implement well-known algorithms for learning those functions in context. For example, when shown 20 examples of the form $(\boldsymbol{x}, \boldsymbol{w}^T\boldsymbol{x})$, where $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^{20}$, the model correctly outputs $\boldsymbol{w}_{\text{test}}^T \boldsymbol{x}_{\text{test}}$ on test input $\boldsymbol{x}_{\text{test}}$ (in noiseless case). Apart from linear regression, they show that for sparse linear regression and shallow neural networks the trained model appears to implement well-known algorithms; and for decision trees, the trained model does better than baselines. Two follow-up works Akyürek et al. [2022] and von Oswald et al. [2022] largely focused on the case of linear regression. Among other things, they showed that transformers with one attention layer learn to implement one step of gradient descent on the linear regression objective with further characterization of the higher number of layers.

Bayesian predictor. An ideal language model (LM) with unlimited training data and compute would learn the pretraining distribution as that results in the smallest loss. Such an LM produces the output by simply sampling from the pretraining distribution conditioned on the input prompt. Such an ideal model is often called *Bayesian predictor*. Many works make the assumption that trained LMs are Bayesian predictors, e.g. Saunshi et al. [2021], Xie et al. [2022], Wang et al. [2023]. Most relevant to the present paper, Akyürek et al. [2022] show that in the MICL setup for linear regression, in the underdetermined setting, namely when the number of examples is smaller than the dimension of the input, the model learns to output the least L_2-norm solution which is the Bayes-optimal prediction. In this paper, we empirically examine how generally transformer LMs follow the Bayesian predictor in the multi-task setting that we introduce.

Prior work has investigated related questions but we are not aware of any extensive empirical verification. E.g., Xie et al. [2022] study a synthetic setup where the pretraining distribution is given by a mixture of hidden Markov models and show that the prediction error of ICL approaches Bayes-optimality as the number of in-context examples approach infinity. In contrast, we test the Bayesian hypothesis for ICL over a wide class of function families and show evidence for equivalence with Bayesian predictor at all prompt lengths. Also closely related, Müller et al. [2022], Hollmann et al. [2023] train transformer models by sampling data from a prior distribution (Prior Fitted Networks), so it could approximate the posterior predictive distribution at inference time. While these works focus on training models to approximate posterior distributions for solving practical tasks (tabular data), our objective is to understand how in-context learning works in transformers and to what extent we can explain it as performing Bayesian Inference on the pre-training distribution.

Our contributions. In brief, our contributions are

1. *A setup for studying ICL for multiple function families*: First, we extend the MICL setup from Garg et al. [2022] to include multiple families of functions. For example, the prompts could be generated from a mixture of these families where the function f is formed from any of the function families (with equal probability) defined by distinct sets of degree-2 monomials of the input. (This is explained in detail in §3.) We call this extended setup HMICL. We experimentally study HMICL and find that high-capacity transformer models can learn in context when given such task mixtures. (We use the term "high-capacity" informally; more precisely, it means that for the task at hand there is a sufficiently large model with the desired property.)

2. *Generalization to new tasks not seen during training in HMICL:* In HMICL setup, we study generalization to new tasks that were not seen during pretraining. We find that when there's sufficient diversity of tasks in pretraining, transformers generalize to new tasks. **Surprisingly, the necessary task diversity required for this generalization is very small** (e.g., for the Monomials problem that we define, we observe that a pretraining distribution induced by only 100 distinct function classes is sufficient for the transformer to generalize to the full distribution induced by all possible function classes (which are about 3^{10} in total). A similar study was made in the concurrent work of Raventós et al. [2023] for the noisy linear regression problem within MICL.

3. *Study of deviations from Bayesian prediction:* Finally, we study deviations from the Bayesian predictor in multitask generalization problems. We study the pretraining inductive bias and find surprising behavior of transformers where **they prefer to generalize to a large set of tasks early in the pretraining and forget this generalization over the course of training**, attempting to fit the pretraining distribution.

2 Background

We first discuss the in-context learning setup for learning function classes as introduced in Garg et al. [2022], which we call **Meta-ICL** or **MICL**. Let $\mathcal{D}_{\mathcal{X}}$ be a probability distribution on $\mathbb{R}^d$. Let $\mathcal{F}$ be a family of functions $f : \mathbb{R}^d \to \mathbb{R}$ and let $\mathcal{D}_{\mathcal{F}}$ be a distribution on $\mathcal{F}$. For simplicity, we often use $f \sim \mathcal{F}$ to mean $f \sim \mathcal{D}_{\mathcal{F}}$. We overload the term function class to encompass both function definition as well as priors on its parameters. (e.g., linear regression with a standard Gaussian prior and a sparse prior will be considered different function classes based on our notation.)

To construct a prompt $P = \big(\boldsymbol{x}_1, f(\boldsymbol{x}_1), \cdots, \boldsymbol{x}_p, f(\boldsymbol{x}_p), \boldsymbol{x}_{p+1}\big)$ of length p, we sample inputs $\boldsymbol{x}_i \sim \mathcal{D}_{\mathcal{X}}$ i.i.d. for $i \in \{1, \cdots p\}$. A transformer-based language model M_θ is trained to predict $f(\boldsymbol{x}_{p+1})$ given P, using the objective: $\min_\theta \mathbb{E}_{f, \boldsymbol{x}_{1:p}} \left[\frac{1}{p+1} \sum_{i=0}^{p} \ell\left(M_\theta(P^i), f(\boldsymbol{x}_{i+1})\right)\right]$, where P^i denotes the sub-prompt containing the first i input-output examples as well as the $(i+1)$-th input, i.e. $\big(\boldsymbol{x}_1, f(\boldsymbol{x}_1), \cdots, \boldsymbol{x}_i, f(\boldsymbol{x}_i), \boldsymbol{x}_{i+1}\big)$ and $\boldsymbol{x}_{1:p} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$. While other choices of the loss function $\ell(\cdot, \cdot)$ are possible, since we study regression problems we use the squared-error loss (i.e., $\ell(y, y') = (y - y')^2$) in accordance with Garg et al. [2022].

At test time, we present the model with prompts P_{test} that were unseen during training with high probability and compute the error when provided k in-context examples: $\texttt{loss@}k = \mathbb{E}_{f, P_{\text{test}}} \left[\ell\left(M_\theta(P^k), f(\boldsymbol{x}_{k+1})\right)\right]$, for $k \in \{1, \cdots, p\}$.

PME. We mentioned earlier that an ideal model would learn the pretraining distribution. This happens when using the cross-entropy loss. Since we use the square loss in the objective definition, the predictions of the model can be computed using the posterior mean estimator (PME) from Bayesian statistics. For each prompt length i we can compute PME by taking the corresponding summand in objective definition above, which will be given by $M_\theta(P^i) = \mathbb{E}_f \left[f(\boldsymbol{x}_{i+1}) \mid P^i\right]$ for all $i \leq p$. This is the optimal solution for prompt P, which we refer to as PME. Please refer to §A.1 for technical details behind this computation.

2.1 Hierarchical Meta-ICL

We generalize the MICL setup, where instead of training transformers from functions sampled from a single function class, we sample them from a mixture of function classes. Formally, we define a mixture of function classes using a set of m function classes $\boldsymbol{\mathcal{F}} = \{\mathcal{F}_1, \cdots, \mathcal{F}_m\}$ and sampling probabilities $\boldsymbol{\alpha} = [\alpha_1, \cdots \alpha_m]^T$ with $\sum_{i=1}^m \alpha_i = 1$. We use $\boldsymbol{\alpha}$ to sample a function class for constructing the training prompt P. We assume the input distribution $\mathcal{D}_{\mathcal{X}}$ to be same for each class $\mathcal{F}_{T_i}$. More concretely, the sampling process for P is defined as:

$$\begin{aligned}
\mathcal{F}_i &\sim \boldsymbol{\mathcal{F}} \text{ s.t. } \mathbb{P}(\mathcal{F} = \mathcal{F}_i) = \alpha_i \\
f &\sim \mathcal{F}_i \\
\boldsymbol{x}_j &\sim \mathcal{D}_{\mathcal{X}}, \forall j \in \{1, \cdots, p\} \\
\text{Finally, } P &= \big(\boldsymbol{x}_1, f(\boldsymbol{x}_1), \cdots \boldsymbol{x}_p, f(\boldsymbol{x}_p), \boldsymbol{x}_{p+1}\big)
\end{aligned}$$

We call this setup **Hierarchical Meta-ICL** or **HMICL**, as there is an additional first step for sampling the function class in the sampling procedure. Note that the MICL setup can be viewed as a special case of HMICL where $m = 1$. In all of our experiments we use uniform mixtures i.e. $\alpha_i = 1/|\mathcal{F}|$ for all i. The HMICL setting presents a more advanced scenario to validate whether the Bayesian inference can be used to explain the behavior of in-context learning in transformers. Further, our HMICL setup is also arguably closer to the in-context learning in practical LLMs which can realize different classes of tasks (sentiment analysis, QA, summarization etc.) depending upon the inputs provided. The PME for the hierarchical case is given by:

$$M_{\theta, \boldsymbol{\mathcal{F}}}(P) = \beta_1 M_{\theta, \mathcal{F}_1}(P) + \ldots + \beta_m M_{\theta, \mathcal{F}_m}(P), \tag{1}$$

where $\beta_i = \alpha_i p_i(P) / p_{\boldsymbol{\mathcal{F}}}(P)$ for $i \leq m$. Probability density $p_i(\cdot)$ is induced by the function class $\mathcal{F}_i$ on the prompts in a natural way, and $p_{\boldsymbol{\mathcal{F}}}(P) = \alpha_i p_i(P) + \cdots + \alpha_m p_m(P)$. Please refer to §A.1 in the Appendix for the derivation. The models are trained with the squared error loss mentioned above.

2.2 Model and training details

We use the decoder-only transformer (TF) architecture Vaswani et al. [2017] as used in the GPT models Radford et al. [2019]. Unless specified otherwise, we use 12 layers, 8 heads, and a hidden size (d_h) of 256 in the architecture for all of our experiments. We use a batch size of 64 and train the model for 500k steps. For encoding the inputs $\boldsymbol{x}_i$'s and $f(\boldsymbol{x}_i)$'s, we use the same scheme as Garg et al. [2022] which uses a linear map $\boldsymbol{E} \in \mathbb{R}^{d_h \times d}$ to embed the inputs $\boldsymbol{x}_i$'s as $\boldsymbol{E}\boldsymbol{x}_i$ and $f(\boldsymbol{x}_i)$'s as $\boldsymbol{E} f_{\text{pad}}(\boldsymbol{x}_i)$, where $f_{\text{pad}}(\boldsymbol{x}_i) = [f(\boldsymbol{x}_i), \mathbf{0}_{d-1}]^T \in \mathbb{R}^d$. In all of our experiments except the ones concerning the Fourier series, we choose $\mathcal{D}_{\mathcal{X}}$ as the standard normal distribution i.e. $\mathcal{N}(0, 1)$, unless specified otherwise. For Fourier series experiments, we choose $\mathcal{D}_{\mathcal{X}}$ to be the uniform distribution $\mathcal{U}(-5, 5)$.

3 Transformers exhibit multi-task generalization in ICL

As stated above, in this section we formulate ICL problems in our newly introduced HMICL setting for the purposes of testing multi-task generalization to out-of-distribution prompts. We work with the degree-2 monomials regression problem, $\mathcal{F}_{\mathcal{S}}^{\text{mon}(2)}$ which is given by a function class where the basis is formed by a feature set $\mathcal{S}$, a subset of degree-2 monomials $\mathcal{S} \subset M = \{(i, j) | \, 1 \leq i, j \leq d\}$. We can then define the feature map $\Phi_{\mathcal{S}}(\boldsymbol{x}) = (x_i x_j)_{(i,j) \in \mathcal{S}}$ and $f(\boldsymbol{x}) = \boldsymbol{w}^T \Phi_{\mathcal{S}}(\boldsymbol{x})$ is a function of this class, where $\boldsymbol{w} \sim \mathcal{N}_{|\mathcal{S}|}(\boldsymbol{0}, \boldsymbol{I})$. We compare the performance of TFs on this class with Ordinary Least Squares (OLS) performed on the feature set $\mathcal{S}$ (OLS$_{\mathcal{S}}$) which is the Bayesian predictor (PME) in this case. We find that the error curves of the TF trained and evaluated on this class follow OLS$_{\mathcal{S}}$ baseline closely for all prompt lengths, on both in- and out-of-distribution evaluation. Note that the above formulation is under the MICL setting, where we only have a single function class corresponding to the feature set $\mathcal{S}$. (The results for this MICL setting are present in §B.2.1 in Appendix, since our focus is HMICL.)

Extending to HMICL setting. For HMICL, we use multiple feature sets $\mathcal{S}_k$'s to define the mixture. Each $\mathcal{S}_k$ defines a function class $\mathcal{F}_{\mathcal{S}_k}^{\text{mon}(2)}$. The pretraining distribution is induced by the uniform distribution $\mathcal{U}(\boldsymbol{\mathcal{F}})$ over a collection of such function classes, $\boldsymbol{\mathcal{F}} = \{\mathcal{F}_{\mathcal{S}_1}^{\text{mon}(2)}, \cdots, \mathcal{F}_{\mathcal{S}_K}^{\text{mon}(2)}\}$, where $\mathcal{S}_k \subset M$. K feature sets $\mathcal{S}_k$'s, each of size D, are chosen at the start of the training and remain fixed. K is the task diversity of the pretraining distribution. To sample a training function for the TF, we first sample a function class $\mathcal{F}_{\mathcal{S}_k}^{\text{mon}(2)}$ with replacement from $\mathcal{U}(\boldsymbol{\mathcal{F}})$ and then sample a function from the chosen class; $f(\boldsymbol{x}) = \boldsymbol{w}^T \mathcal{S}_k(\boldsymbol{x})$, where $\boldsymbol{w} \sim \mathcal{N}_D(\boldsymbol{0}, \boldsymbol{I})$. Our aim is to check if TF trained on $\mathcal{U}(\boldsymbol{\mathcal{F}})$ can generalize to the full distribution of all function classes (for feature sets of size D) by evaluating its performance on function classes corresponding to feature sets $\mathcal{S}' \notin \{\mathcal{S}_1, \cdots, \mathcal{S}_K\}$.

Experimental Setup. We choose $D = d = 10, p = 124$. Note that the total number of degree-2 monomials $= M_{tot} = \binom{d}{2} + \binom{d}{1} = 45 + 10 = 55$; and the total number of distinct feature sets $\mathcal{S}_k$'s (and hence function classes), $F_{tot} = \binom{M_{tot}}{D} = \binom{55}{10} \approx 3^{10}$. We train various models for different task diversities; $K \in \{10, 20, 40, 100, 500, 1000, 5000\}$. We evaluate on a batch of $B = 1280$ functions in two settings: **(a) In-Distribution (ID)** – test functions formed using randomly chosen function classes from the pretraining distribution; **(b) Out-of-Distribution (OOD)** – Test functions formed using randomly chosen function classes not in the pretraining distribution.

Baselines. We compare the performance of multi-task transformer models with the following baselines: **1. OLS$_{\mathcal{S}}$**: Here, we perform OLS on the basis formed by the gold feature set $\mathcal{S}$, which was used to define the function in the prompt that we wish to evaluate. This will correspond to an upper bound on the performance as at test time the transformer model has no information about the correct basis. **2. OLS$_{\Phi_M}$**: Here, OLS is performed on the basis formed by all degree-2 monomials $\Phi_M(\boldsymbol{x})$ for an input $\boldsymbol{x}$. Hence, this baseline can generalize to any of the feature set $\mathcal{S}$. However, since all degree-2 monomial features are considered by this baseline, it would require a higher number of input-output examples (equal to M_{tot}) for the problem to be fully determined. **3. Lasso$_{\Phi_M}$**: Similar to OLS$_{\Phi_M}$, we operate on all degree-2 monomial features, but instead of OLS we perform Lasso with $\alpha = 0.1$. It should also generalize to arbitrary feature sets $\mathcal{S}$, however, Lasso can take advantage of the fact that $|\mathcal{S}| = D \ll M_{tot}$; hence should be more efficient than OLS$_{\Phi_M}$. **4. (BP$_{\text{proxy}}$)**: This is the TF trained on the full distribution induced by all possible function families. We use it as a proxy for the Bayesian predictor on the full distribution since the computation of the exact predictor is expensive in this setting.

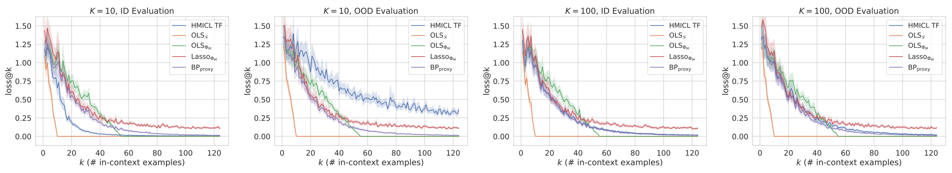


Figure 1: **Multi-task generalization results for Monomials problem**. ID and OOD evaluation for $K = 10, 100$ is presented. As task diversity (K) increases, the model starts behaving like Lasso$_{\Phi_M}$ and BP$_{\text{proxy}}$ and its ID and OOD losses become almost identical, i.e. it generalizes to OOD.

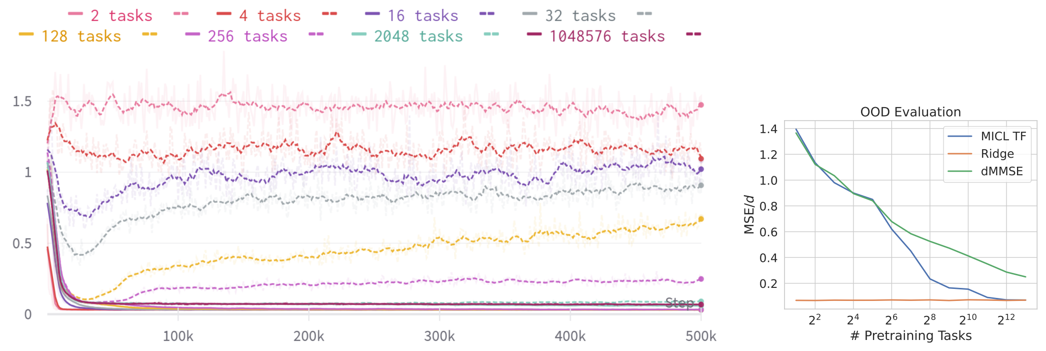


Figure 2: **Left:** Evolution of ID (solid lines) and OOD (dashed lines) losses during pretraining for representative task diversities. Task diversities $\{2^7 \cdots 2^{11}\}$ represent the **Gaussian forgetting region**. The moving average (over 10 training steps) of the losses is plotted for smoothing. **Right:** OOD loss given the full prompt length of 15 for the final checkpoint of models trained on various task diversities. Task diversities $\{2^7 \cdots 2^{11}\}$ represent the **transition region**.

Results. From the plots in Figure 1, we observe that while for small values of K (viz. $K = 10$), the OOD generalization is poor, as we move to higher values of K (viz. $K = 100$), the models start to approach the performance of OLS$_{\Phi_M}$ and eventually Lasso$_{\Phi_M}$ on unseen $\mathcal{S}'$s. Further, they also start behaving like BP$_{\text{proxy}}$. This is surprising since $100 \ll F_{tot} = 3^{10}$, i.e. TFs can perform multi-task generalization even when they are trained only on a sample of the full distribution corresponding to a small number of function classes! However, this improvement in OOD performance comes at the cost of ID performance as task diversity (K) increases. Eventually, at larger K, both ID and OOD performances are identical. These observations are particularly interesting since the models learn to generalize to function classes out of the pre-training distribution and hence deviate from the Bayesian predictor of the pretraining distribution which would lack such generalization and fit the pre-training distribution instead. Plots for more task diversities are in §C.1. We observe similar results for another family of function classes coming from Fourier series (details of these are in Appendix §C.2).

In a concurrent work Raventós et al. [2023] also present a multi-task setting within MICL where a set of weight vectors define the pretraining distribution for the Noisy Linear Regression problem. Since we work with HMICL, our setting is more general; moreover, generalization to new function classes in our setting happens in a similar way as generalization to new tasks in Raventós et al. [2023]. They emphasized deviation from the Bayesian predictor. What leads to these deviations? To understand this, in the next section we study pretraining inductive bias of transformers.

4 ICL Transformer first generalizes then memorizes during pretraining

In the previous section we observed deviations from the Bayesian predictor in multitask generalization. To investigate this we study the pretraining dynamics of transformers. We observe a very interesting phenomenon (which we term "forgetting") from multi-task experiments: *For certain task diversities, during pretraining, HMICL Transformer first generalizes (fits the full distribution) and later forgets it and memorizes (fits the pretraining distribution).*

The 'forgetting' phenomenon is general and occurs in our HMICL experiments in §3. However, here we focus on the the Noisy Linear Regression problem from Raventós et al. [2023] since forgetting is the cleanest in this setting. We briefly mention the problem setup and display the evidence for forgetting during pretraining, followed by its relation to the agreement of HMICL Transformer with the Bayesian predictors on the pretraining and full distributions.

Problem Setup. We follow the Noisy Linear Regression (NLR) setup from Raventós et al. [2023]: $d = 8, p = 15$. (For details, see §C.3.) The pretraining distribution (PT$_{\text{dist.}}$) is induced by the uniform distribution on a fixed set of tasks (weight vectors). Several models are trained, one per task diversity $K \in \{2^1, 2^2, \cdots 2^{20}\}$. The full distribution of weight vectors is standard normal. (Hence we use the term "Gaussian distribution" to refer to the full distribution (FG$_{\text{dist.}}$).) To form a function f for training, we randomly choose a weight vector $\boldsymbol{w}$ from the pretraining distribution and define $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}_d(0, \sigma^2 = 0.25)$.

Evidence of forgetting and agreement with Bayesian predictors. As we did in §3, we evaluate TF on tasks from both in- and out-of-pretraining distribution, where the tasks used to construct the test function come from pretraining distribution or the standard Gaussian distribution respectively; corresponding losses are called ID (Pretrain test) loss and OOD (Gaussian) loss. We also plot the Bayesian predictors for both pretraining (dMMSE) and full (Gaussian) distribution (Ridge regression) as defined in Raventós et al. [2023]. In Figure 2 (left) we plot the evolution during pretraining of ID and OOD losses for representative task diversities (more details in §C.3) ID loss ≈ 0 for all task diversities. We group them into the following 4 categories based on OOD loss and describe the most interesting one in detail (full classification in §C.3): **(1)** 2^1 to 2^3: *no generalization; no forgetting*; **(2)** 2^4 to 2^6: *some generalization; no forgetting*; **(3)** 2^7 to 2^{11}: *full generalization and forgetting* – OOD loss improves, reaches a minima t_{min}, at which it is same as ID loss, then it worsens. At t_{min}, OOD loss agrees with Ridge, then gradually deviates from it and at t_{end} (end of pretraining), it is in between dMMSE and Ridge. We refer to this group of task diversities as the "Gaussian forgetting region" since the model generalizes to the full (Gaussian) distribution over tasks at t_{min} but forgets it by t_{end}; **(4)** 2^{12} to 2^{20}: *full generalization; no forgetting*.

The agreement of TF in OOD evaluation with Ridge or dMMSE (in terms of loss and implied weights) as mentioned above is shown in §C.3. Figure 2 (right) plots the OOD loss given the full prompt length of 15 for the final checkpoint of models trained for various task diversities. As can be seen, smaller task diversities (up to 2^6) agree with dMMSE (Bayesian predictor on PT$_{\text{dist.}}$), and larger task diversities (from 2^{12} onwards) agree with Ridge regression (Bayesian predictor on FG$_{\text{dist.}}$). (This observation was originally made by Raventós et al. [2023] and we present it for completeness.) Intermediate task diversities (2^7 to 2^{11}) agree with neither of the two and we term them collectively as the **transition region**. We note that both **the Gaussian forgetting region and the transition region consist of the same set of task diversities** viz. $\{2^7, \cdots 2^{11}\}$. The phenomenon of forgetting provides an interesting contrast to grokking literature, e.g. Nanda et al. [2023], where they find that the model first memorizes and then generalizes (which, on the surface, is the opposite of what we observe). The extent of forgetting is directly proportional to the input dimension (d) and is robust to changes in hyperparameters (details, in section §C.3).

Simplicity bias. Simplicity bias is the tendency of machine learning algorithms to prefer simpler hypotheses among those consistent with the data, which has been suggested as the basis of the success of neural networks. There are many notions of simplicity [Mingard et al., 2023, Goldblum et al., 2023]. The phenomenon of forgetting can possibly be explained via the perspective of simplicity bias. (details, in section §C.3)

5 Conclusion

In this paper, we extended the MICL setting from the literature to HMICL setting and empirically showed that it gives rise to interesting and surprising observations depicting deviation from Bayesian inference on the pretraining distribution. In particular, for the multi-task setting we identified how transformers generalize to new tasks, deviating from Bayesian predictor on the pretraining distribution. We also observed that transformers have a pretraining inductive bias in HMICL that leads to generalization on the full distribution followed by memorization of the pretraining distribution. There are many interesting directions for future work. Much more remains to be done to determine how extensively transformers mimic the Bayesian predictor. The intriguing forgetting phenomenon needs to be better understood. How is it related to pretraining simplicity bias? Finally, we treated transformers as black boxes: opening the box and uncovering the underlying mechanisms transformers use to do Bayesian prediction would be very interesting.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *CoRR*, abs/2211.15661, 2022. doi: 10.48550/arXiv.2211.15661. URL https://doi.org/10.48550/arXiv.2211.15661.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Stephanie C. Y. Chan, Ishita Dasgupta, Junkyung Kim, Dharshan Kumaran, Andrew K. Lampinen, and Felix Hill. Transformers generalize differently from information stored in context vs in weights. *CoRR*, abs/2210.05675, 2022. doi: 10.48550/arXiv.2210.05675. URL https://doi.org/10.48550/arXiv.2210.05675.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning, 2023.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.

Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *CoRR*, abs/2304.05366, 2023. doi: 10.48550/arXiv.2304.05366. URL https://doi.org/10.48550/arXiv.2304.05366.

Michael Hahn and Navin Goyal. A theory of emergent in-context learning as implicit structure induction. *CoRR*, abs/2303.07971, 2023. doi: 10.48550/arXiv.2303.07971. URL https://doi.org/10.48550/arXiv.2303.07971.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(09):5149–5169, sep 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35, 2023. doi: 10.1145/3560815. URL https://doi.org/10.1145/3560815.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States, July 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.201. URL `https://aclanthology.org/2022.naacl-main.201`.

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? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates, December 2022b. Association for Computational Linguistics. URL `https://aclanthology.org/2022.emnlp-main.759`.

Chris Mingard, Henry Rees, Guillermo Valle Pérez, and Ard A. Louis. Do deep neural networks have an inbuilt occam's razor? *CoRR*, abs/2304.06670, 2023. doi: 10.48550/arXiv.2304.06670. URL `https://doi.org/10.48550/arXiv.2304.06670`.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=KSugKcbNf9`.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=9XFSbDPmdW`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf*, 1(8):9, 2019.

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

Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=vVjIW3sEc1s`.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. 2022.

Xinyi Wang, Wanrong Zhu, and William Yang Wang. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *CoRR*, abs/2301.11916, 2023. doi: 10.48550/arXiv.2301.11916. URL `https://doi.org/10.48550/arXiv.2301.11916`.

Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.167. URL `https://aclanthology.org/2022.naacl-main.167`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL `https://aclanthology.org/2020.emnlp-demos.6`.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=RdJVFCHjUMI`.

Contents

A Technical Details

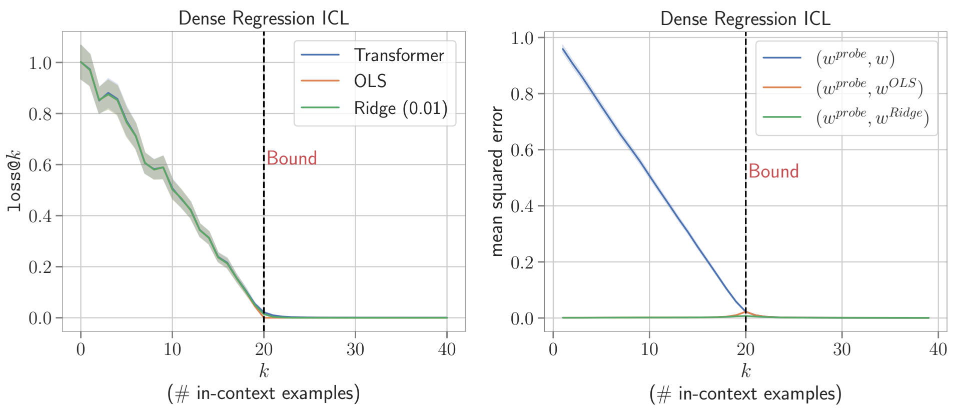
A.1 PME Theoretical Details

We mentioned earlier that an ideal LM would learn the pretraining distribution. This happens when using the cross-entropy loss. Since we use the square loss in the ICL training objective, the predictions of the model can be computed using the posterior mean estimator (PME) from Bayesian statistics. For each prompt length i we can compute PME by taking the corresponding summand in the ICL training objective

$$\begin{aligned}\min_\theta \mathbb{E}_{f,\boldsymbol{x}_{1:i}}\, \ell\left(M_\theta(P^i), f(\boldsymbol{x}_{i+1})\right) &= \min_\theta \mathbb{E}_{f,P^i}\, \ell\left(M_\theta(P^i), f(\boldsymbol{x}_{i+1})\right)\\ &= \min_\theta \mathbb{E}_{P^i}\, \mathbb{E}_f\left[\ell\left(M_\theta(P^i), f(\boldsymbol{x}_{i+1})\right) \mid P^i\right]\\ &= \mathbb{E}_{P^i}\, \min_\theta \mathbb{E}_f\left[\ell\left(M_\theta(P^i), f(\boldsymbol{x}_{i+1})\right) \mid P^i\right].\end{aligned}$$

The inner minimization is seen to be achieved by $M_\theta(P^i) = \mathbb{E}_f\left[f(\boldsymbol{x}_{i+1}) \mid P^i\right]$. This is the optimal solution for prompt P^i and what we refer to as PME.

PME for a task mixture. We describe the PME for a mixture of function classes. For simplicity we confine ourselves to mixtures of two function classes; extension to more function classes is analogous.



(a) (b)

Figure 3: Results on the Dense Regression ($\mathcal{F}_{\text{DR}}$) function class.

Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two function classes specified by probability distributions $\mathcal{D}_{\mathcal{F}_1}$ and $\mathcal{D}_{\mathcal{F}_2}$, resp. As in the single function class case, the inputs $\boldsymbol{x}$ are chosen i.i.d. from a common distribution $\mathcal{D}_{\mathcal{X}}$. For $\alpha_1, \alpha_2 \in [0,1]$ with $\alpha_1 + \alpha_2 = 1$, an (α_1, α_2)-mixture $\boldsymbol{\mathcal{F}}$ of $\mathcal{F}_1$ and $\mathcal{F}_2$ is the meta-task in which the prompt $P = \big(\boldsymbol{x}_1, f(\boldsymbol{x}_i), \cdots, \boldsymbol{x}_p, f(\boldsymbol{x}_p), \boldsymbol{x}_{p+1}\big)$ is constructed by first picking task $\mathcal{F}_i$ with probability α_i for $i \in \{1,2\}$ and then picking $f \sim \mathcal{D}_{\mathcal{F}_i}$. Thus $p_{\boldsymbol{\mathcal{F}}}(f) = \alpha_1 p_{\mathcal{F}_1}(f) + \alpha_2 p_{\mathcal{F}_2}(f)$, where $p_{\mathcal{F}}(\cdot)$ is the probability density under function class $\mathcal{F}$ which defines $\mathcal{D}_{\mathcal{F}}$. For conciseness in the following we use $p_1(\cdot)$ for $p_{\mathcal{F}_1}(\cdot)$ etc. Now recall that PME for function class $\mathcal{F}$ is given by

$$M_{\theta,\mathcal{F}}(P) = \mathbb{E}_{f\sim\mathcal{D}_{\mathcal{F}}}\left[f(\boldsymbol{x}_{p+1}) \mid P\right] = \int p_{\mathcal{F}}(f|P)\, f(x)\, \mathrm{d}f. \tag{2}$$

We would like to compute this in terms of PMEs for $\mathcal{F}_1$ and $\mathcal{F}_2$. To this end, we first compute

$$\begin{aligned} p_{\boldsymbol{\mathcal{F}}}(f|P) &= \frac{p_{\boldsymbol{\mathcal{F}}}(P|f)p_{\boldsymbol{\mathcal{F}}}(f)}{p_{\boldsymbol{\mathcal{F}}}(P)} = \frac{p(P|f)p_{\boldsymbol{\mathcal{F}}}(f)}{p_{\boldsymbol{\mathcal{F}}}(P)} = \frac{p(P|f)}{p_{\boldsymbol{\mathcal{F}}}(P)}\left[\alpha_1 p_1(f) + \alpha_2 p_2(f)\right] \\ &= \frac{\alpha_1 p_1(P)}{p_{\boldsymbol{\mathcal{F}}}(P)} \frac{p(P|f)p_1(f)}{p_1(P)} + \frac{\alpha_2 p_2(P)}{p_{\boldsymbol{\mathcal{F}}}(P)} \frac{p(P|f)p_2(f)}{p_2(P)} \\ &= \frac{\alpha_1 p_1(P)}{p_{\boldsymbol{\mathcal{F}}}(P)} p_1(f|P) + \frac{\alpha_2 p_2(P)}{p_{\boldsymbol{\mathcal{F}}}(P)} p_2(f|P) \\ &= \beta_1\, p_1(f|P) + \beta_2\, p_2(f|P), \end{aligned}$$

where $\beta_1 = \frac{\alpha_1 p_1(P)}{p_{\boldsymbol{\mathcal{F}}}(P)}$ and $\beta_2 = \frac{\alpha_2 p_2(P)}{p_{\boldsymbol{\mathcal{F}}}(P)}$. Plugging this in equation 2 we get

$$M_{\theta,\boldsymbol{\mathcal{F}}}(P) = \beta_1 \int p_1(f|P)\, f(x)\, \mathrm{d}f + \beta_2 \int p_2(f|P)\, f(x)\, \mathrm{d}f = \beta_1 M_{\theta,\mathcal{F}_1}(P) + \beta_2 M_{\theta,\mathcal{F}_2}(P). \tag{3}$$

A.2 Experimental Setup

We use Adam optimizer Kingma and Ba [2015] to train our models. Our experiments were conducted on a system comprising 32 NVIDIA V100 16GB GPUs. The cumulative training time of all models for this project was $\sim$ 15,000 GPU hours. While reporting the results, the error is averaged over 1280 prompts and shaded regions denote a 90% confidence interval over 1000 bootstrap trials.

We adapt Garg et al. [2022] code-base for our experiments. We use PytorchPaszke et al. [2019] and Huggingface TransformersWolf et al. [2020] libraries to implement the model architecture and training procedure. For the baselines against which we compare transformers, we use `scikit-learn`'s [1] implementation of OLS, Ridge and Lasso.

[1]https://scikit-learn.org/stable/index.html

B MICL Experiments

B.1 Dense Regression ($\mathcal{F}_{\text{DR}}$).

This represents the simplest case of linear regression as studied in Garg et al. [2022], Akyürek et al. [2022], von Oswald et al. [2022], where the prior on $\boldsymbol{w}$ is the standard Gaussian i.e. $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I})$. We are particularly interested in the underdetermined region i.e. $k < d$. Gaussian prior enables explicit PME computation: both PME and maximum a posteriori (MAP) solution agree and are equal to the minimum L_2-norm solution of the equations forming the in-context examples, i.e. $\min_{\boldsymbol{w}} \|\boldsymbol{w}\|_2$ s.t. $\boldsymbol{w}^T \boldsymbol{x}_i = f(\boldsymbol{x}_i), \forall i \leq k$. Standard Ordinary Least Squares (OLS) solver returns the minimum L_2-norm solution, and is thus the PME and MAP.

Experimental Details. We train transformer-based model with $d = 20$ and $p = 40$. Additionally, we also extract the implied weights $\boldsymbol{w}^{\text{probe}}$ from the trained models when given a prompt P following Akyürek et al. [2022] by generating model's predictions $\{y'_i\}$ on the test inputs $\{\boldsymbol{x}'_i\}_{i=1}^{2d} \sim \mathcal{D}_{\mathcal{X}}$ and then solving the system of equations to recover $\boldsymbol{w}^{\text{probe}}$. We then compare the implied weights $\boldsymbol{w}^{\text{probe}}$ with the ground truth weights $\boldsymbol{w}$ as well as the weights extracted from different baselines to better understand the inductive biases exhibited by these models during in-context learning.

Results. The results for dense regression have been already covered in Akyürek et al. [2022] and for completeness, we provide them in Figure 3. We observe that Transformer follows the Bayesian predictor OLS, i.e. the Transformer's loss as well as implied weights agree with those found by OLS.

B.2 Experiments with non-linear function classes

For multi-task generalization, we experiment with non-linear function classes, i.e. where the output $f(\boldsymbol{x})$ is a non-linear function of the input $\boldsymbol{x}$. Particularly, we consider the function classes of the form $\mathcal{F}_\Phi = \left\{f(\cdot;\Phi) | f(\boldsymbol{x};\Phi) = \boldsymbol{w}^T\Phi(\boldsymbol{x}), \boldsymbol{w} \in \mathbb{R}^\Delta\right\}$, where $\Phi : \mathbb{R}^d \to \mathbb{R}^\Delta$ maps the input vector $\boldsymbol{x}$ to an alternate feature representation. This corresponds to learning the mapping $\Phi(\boldsymbol{x})$ and then performing linear regression on top of it. Under the assumption of a standard Gaussian prior on $\boldsymbol{w}$, the PME for the dense regression can be easily extended for $\mathcal{F}_\Phi$: $\min_{\boldsymbol{w}} \|\boldsymbol{w}\|_2$, s.t. $\boldsymbol{w}^T\Phi(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ for $i \in \{1, \cdots, p\}$.

B.2.1 Degree-2 Monomial Basis Regression

We define degree-2 monomials regression problem, $\mathcal{F}_{\mathcal{S}}^{\text{mon(2)}}$ which is given by a function class where the basis is formed by a feature set $\mathcal{S}$, a subset of degree-2 monomials $\mathcal{S} \subset M = \{(i,j) | \, 1 \leq i, j \leq d\}$. We can then define the feature map $\Phi_{\mathcal{S}}(\boldsymbol{x}) = (x_i x_j)_{(i,j)\in\mathcal{S}}$ and $f(\boldsymbol{x}) = \boldsymbol{w}^T\Phi_{\mathcal{S}}(\boldsymbol{x})$ is a function of this class, where $\boldsymbol{w} \sim \mathcal{N}_{|\mathcal{S}|}(\boldsymbol{0}, \boldsymbol{I})$. We compare the performance of TFs on this class with OLS performed on the feature set $\mathcal{S}$ (OLS$_{\mathcal{S}}$) which is the Bayesian predictor (PME) in this case. We find that the error curves of the TF trained and evaluated on this class follow OLS$_{\mathcal{S}}$ baseline closely for all prompt lengths, on both in- and out-of-distribution evaluation. Note that the above formulation is under the MICL setting, where we only have a single function class corresponding to the feature set $\mathcal{S}$. We experiment with $d = 20$, with the prompt length $p = 290$ and $|\mathcal{S}| = 20$.

Baselines. We use OLS fitted to the following bases as baselines: feature set $\mathcal{S}$ (OLS$_{\mathcal{S}}$), all degree-2 monomials i.e., Φ_M (OLS$_{\Phi_M}$), and to a basis of all polynomial features up to degree-2 (OLS$_{\text{poly.(2)}}$). We also compare Lasso ($\alpha = 0.01$) fitted to all degree-2 monomials i.e., Φ_M (Lasso$_{\Phi_M}$) as a baseline.

Results. In Figure 4, we show the In-Distribution (ID) evaluation results for the $\mathcal{F}_{\mathcal{S}}^{\text{mon(2)}}$ experiments. Here, the test prompts contain functions formed by $\mathcal{S}$ (the same feature set used during training). We observe that Transformers closely follow OLS$_{\mathcal{S}}$. The increasing order of performance (decreasing `loss@`k for $k \geq |\mathcal{S}|$) of different solvers is: OLS$_{\text{poly.(2)}}$ $\leq$ OLS$_{\Phi_M}$ $<$ Lasso$_{\Phi_M}$ $<$ Transformers $<$ OLS$_{\mathcal{S}}$. Transformer's squared error takes a little longer than OLS$_{\mathcal{S}}$ to converge. Lasso$_{\Phi_M}$ is able to take the advantage of sparsity of the problem and is hence better than both OLS$_{\Phi_M}$ and OLS$_{\text{poly.(2)}}$, which respectively converge at $k = 210$ and $k = 231$[2]. We also conduct an Out-of-Distribution

[2] 210 and 231 are the sizes of the bases to which OLS$_{\Phi_M}$ and OLS$_{\text{poly.(2)}}$ are fitted. Hence, they converge right when the problem becomes determined in their respective bases.

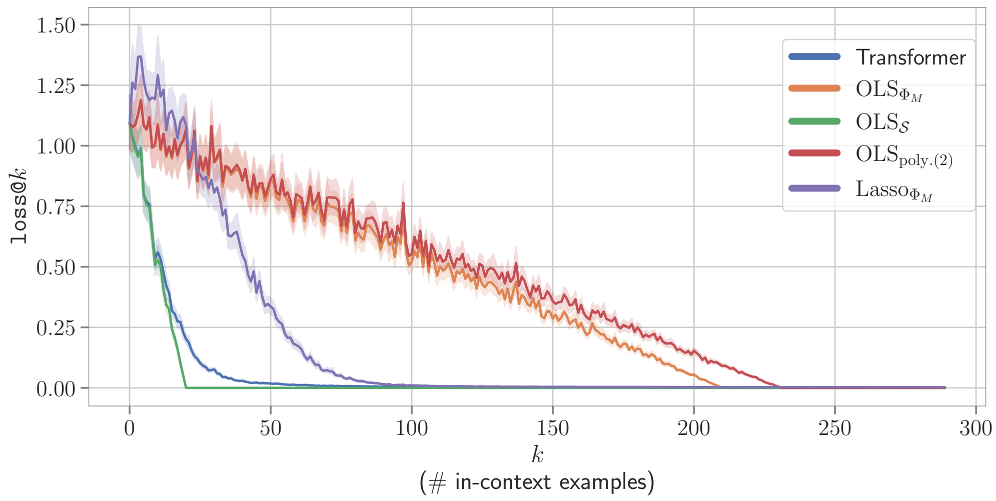


Figure 4: **In-Distribution evaluation results on $\mathcal{F}_{\mathcal{S}}^{\mathbf{mon(2)}}$ sub-family of degree-2 monomial basis regression.** Evaluation of transformer on prompts generated using the same $\mathcal{S}$ used during training.

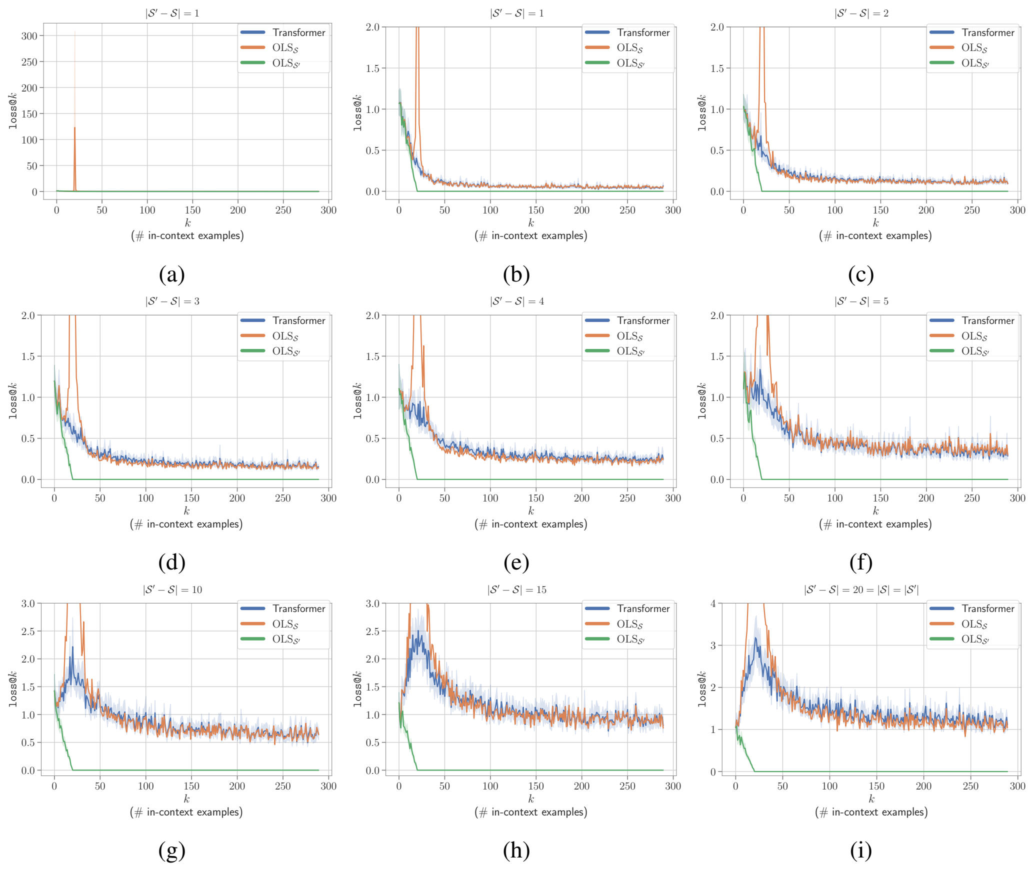


Figure 5: **Out-of-Distribution evaluation results on $\mathcal{F}_{\mathcal{S}}^{\mathbf{mon(2)}}$ sub-family of degree-2 monomial basis regression.** Evaluation of transformer trained on prompts generated using $\mathcal{S}'$, where $\mathcal{S}'$ contains n degree-2 monomials not present in $\mathcal{S}$ that was used during training. We show results for different values of n.

(OOD) evaluation for $\mathcal{F}_{\mathcal{S}}^{\text{mon}(2)}$, whose results are shown in Figure 5. Here, we generate prompts from a feature set $\mathcal{S}' \subset \Phi_M$ of the same size as $\mathcal{S}$ but differing from $\mathcal{S}$ in n degree-2 terms, i.e. $|\mathcal{S}' - \mathcal{S}| = n$. We show the results for different values of n. Figure 5a shows the OLS$_{\mathcal{S}}$ undergoes a steep rise in errors momentarily at $k = |\mathcal{S}|$ (double descent). Figure 5b zooms into the lower error region of Figure 5a where we notice that Transformer mimics OLS$_{\mathcal{S}}$, while OLS$_{\mathcal{S}'}$ is the best-performing baseline (since it fits to the $\mathcal{S}'$ basis used to construct the prompts). Transformer does not undergo double descent (for $n = 1$) and is hence momentarily better than OLS$_{\mathcal{S}}$ at $k = |\mathcal{S}|$. Similar plots are shown for $n \in \{2, 3, 4, 5, 10, 15, 20\}$. As n increases, the height of OLS$_{\mathcal{S}}$ peak increases and the Transformer also starts to have a rise in errors at $k = |\mathcal{S}|$. For $n = 20$, $\mathcal{S}'$ and $\mathcal{S}$ have nothing in common, and Transformer still follows OLS$_{\mathcal{S}}$ (OLS fitted to the training basis $\mathcal{S}$). As mentioned under §B.2, when the prior on weights $\boldsymbol{w}$ is Gaussian, the PME is the minimum L_2-norm solution. For $\mathcal{F}_{\mathcal{S}}^{\text{mon}(2)}$, that solution is given by OLS$_{\mathcal{S}}$. Therefore, the results suggest that the transformer is computing PME. In summary, transformers closely follow OLS$_{\mathcal{S}}$ in this set-up, and more so on the OOD data, where they even surpass OLS$_{\mathcal{S}}$'s performance when it experiences double descent.

B.2.2 Fourier Series

A Fourier series is an expansion of a periodic function into a sum of trigonometric functions. One can represent the Fourier series using the sine-cosine form given by:

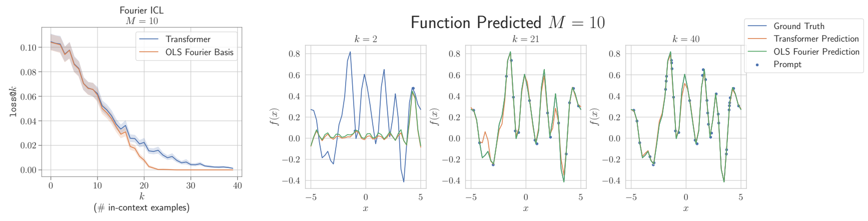
$$f(x) = a_0 + \sum_{n=1}^{N} a_n \cos(n\pi x/L) + \sum_{n=1}^{N} b_n \sin(n\pi x/L)$$

where, $x \in [-L, L]$, and a_0, a_n's and b_n's are known as Fourier coefficients and $\cos n\pi/L$ and $\sin n\pi/L$ define the frequency n components. We can define the function class $\mathcal{F}_{\Phi_N}^{\text{fourier}}$ by considering Φ as the Fourier feature map i.e. $\Phi_N(x) = [1, \cos(\pi x/L), \cdots, \cos(N\pi x/L), \sin(\pi x/L), \cdots, \sin(N\pi x/L)]^T$, and $\boldsymbol{w}$ as Fourier coefficients: $\boldsymbol{w} = [a_0, a_1, \cdots, a_N, b_1, \cdots, b_N]$. Hence, $\Phi_N(x) \in \mathbb{R}^d$ and $\boldsymbol{w} \in \mathbb{R}^d$, where $d = 2N + 1$.

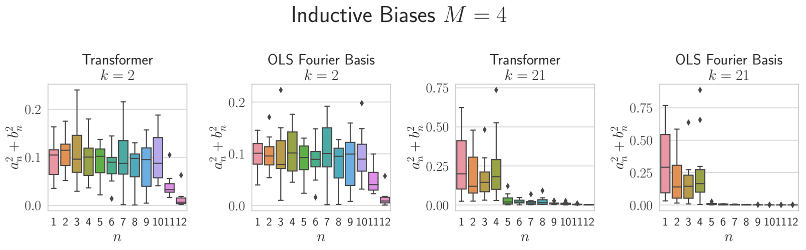
For training transformers to in-context-learn $\mathcal{F}_{\Phi_N}^{\text{fourier}}$, we fix a value of N and sample functions $f \in \mathcal{F}_{\Phi_N}^{\text{fourier}}$ by sampling the Fourier coefficients from the standard normal distribution i.e. $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I})$. (Since we only have a single function class, viz. $\mathcal{F}_{\Phi_N}^{\text{fourier}}$, this is the MICL setting.) We consider the inputs to be scalars, i.e. $x_i \in [-L, L]$ and we sample them i.i.d. from the uniform distribution on the domain: $x_i \sim \mathcal{U}(-L, L)$. In all of our experiments, we consider $N = 10$ and $L = 5$. At test time we evaluate on $\mathcal{F}_{\Phi_M}^{\text{fourier}}$ for $M \in [1, 10]$, i.e. during evaluation we also prompt the model with functions with different maximum frequency as seen during training. As a baseline, we use OLS on the Fourier features (denoted as OLS Fourier Basis) which is the PME.

Measuring inductive biases. Once we train a transformer-based model to in-context learn $\mathcal{F}_{\Phi_N}^{\text{fourier}}$, how can we investigate the inductive biases that the model learns to solve the problem? We would like to answer questions such as, when prompted with k input-output examples what are the prominent frequencies in the function simulated by the model, or, how do these exhibited frequencies change as we change the value of k? We start by sampling in-context examples $(x_1, f(x_1), \cdots x_k, f(x_k))$, and given the context obtain the model's predictions on a set of m test inputs $\{x'_i\}_{i=1}^m$, i.e. $y'_i = M_\theta\left((x_1, f(x_1), \cdots x_k, f(x_k), x'_i)\right)$. We can then perform Discrete Fourier Transform (DFT) on $\{y'_1, \cdots, y'_m\}$ to obtain the Fourier coefficients of the function output by M, which we can analyze to understand the dominant frequencies.

Results. The results of our experiments concerning the Fourier series are provided in Figure 6. Transformers obtain `loss@`k values close to the OLS Fourier Basis baseline (Figure 6a) indicating at least for the smaller prompt lengths the model is able to simulate the behavior of the ideal predictor (PME). These plots use 12-layer transformers to obtain results, but we also investigate if bigger models help. Figure 7 plots bigger models with 18 and 21 layers where the agreement with PME is much better. Since the inputs x_i, in this case, are scalars, we can visualize the functions learned in context by transformers. We show one such example for a randomly selected function $f \sim \mathcal{F}_{\Phi_M}^{\text{fourier}}$ for prompting the model in Figure 6b. As can be observed, the functions predicted by both the transformer and baseline have a close alignment, and both approach the ground truth function f as more examples are provided. Finally, we visualize the distribution of the frequencies for the predicted functions in Figure 6c. For a value of M, we sample 10 different functions and provide k in-context



(a) (b)



(c)

Figure 6: Effectiveness of ICL in transformers for Fourier series family of functions. **Top left**: loss@k values for transformer and OLS Fourier Basis baseline. **Top Right**: Visualizing the functions simulated by the transformer and the OLS Fourier Basis. **Bottom**: Measuring the frequencies of the simulated function by the transformer and the baseline.

examples to the model to extract the frequencies of the predicted functions using the DFT method. As can be observed, when provided with fewer in-context examples ($k = 2$) both Transformer and the baseline predict functions with all the 10 frequencies (indicated by the values of $a_n^2 + b_n^2$ in a similar range for $n \in [1, 10]$), but as more examples are provided they begin to recognize the gold maximum frequency (i.e. $M = 4$). The function visualizations for the transformer and Fourier OLS baseline for different combinations of M and k are provided in Figure 9. We have observations consistent with Figure 6b, where the function outputs of the transformer and the baseline align closely. Similarly, in Figure 8, we present the distribution of frequencies in the predicted functions for the two methods and again observe consistent findings. This suggests that the transformers are following the Bayesian predictor - OLS Fourier Basis.

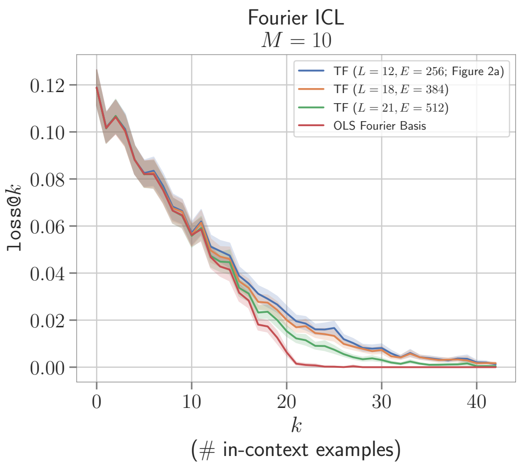


Figure 7: **Bigger models achieve better results on the Fourier Series task.** Plotting the squared error (averaged over 1280 prompts) for bigger transformer (TF) models trained for 500k steps on the Fourier Series task. Training setup is the same as used for the model plotted in Figure 2a (Section 3.2.1), which is also plotted here for comparison. L and E denote the number of layers and embedding size for TF models respectively.

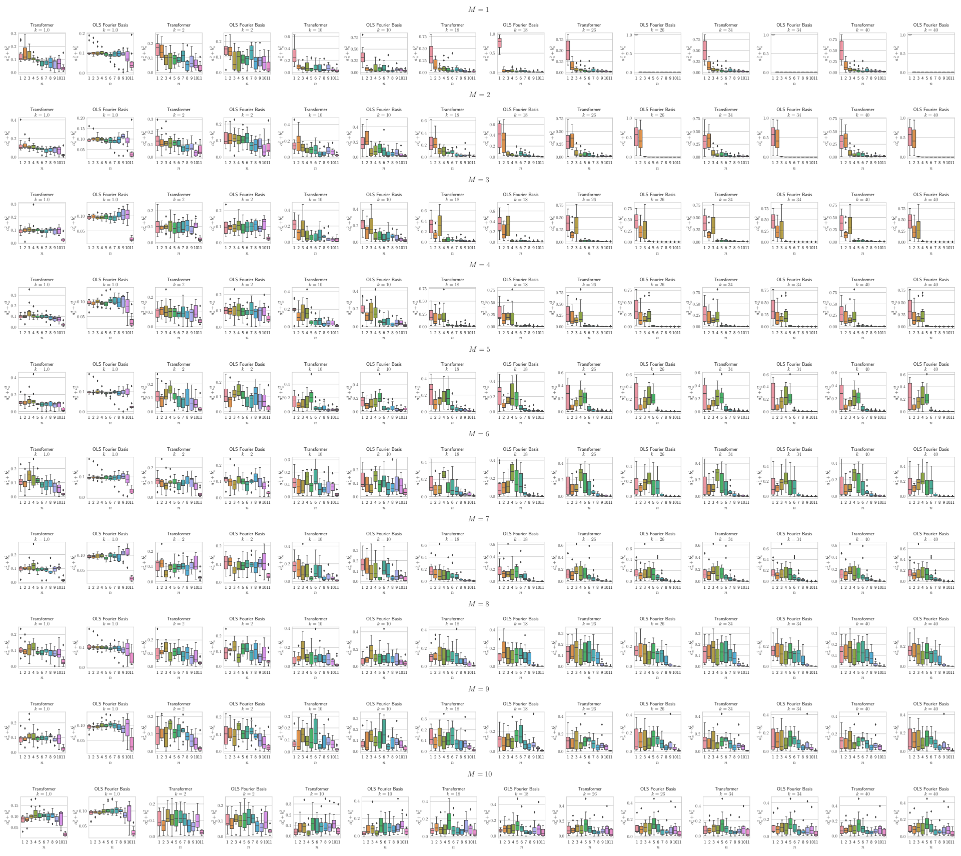
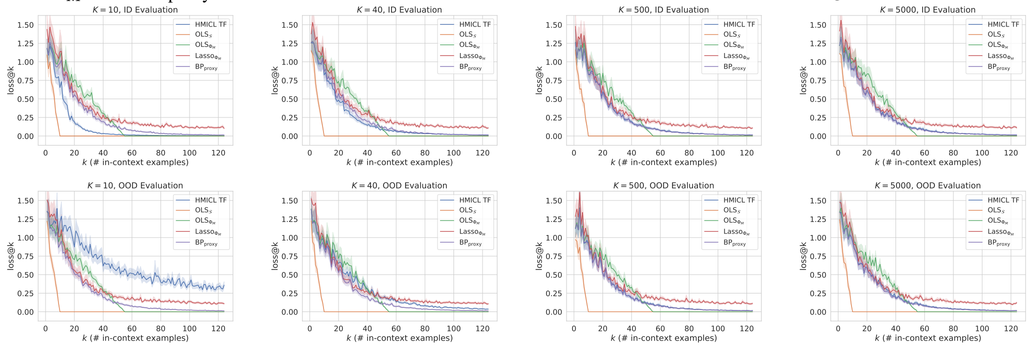


Figure 8: Measuring the frequencies of the simulated function by the transformer and the baseline for different values of M (maximum frequency) and k (number of in-context examples)

Table 1: **Multi-task generalization results for Monomials problem**. The first row is ID evaluation, second row is OOD evaluation. As task diversity (K) increases, the model starts behaving like Lasso_{Φ_M} and BP_{proxy}, and its ID and OOD losses become almost identical, i.e. it generalizes to OOD.



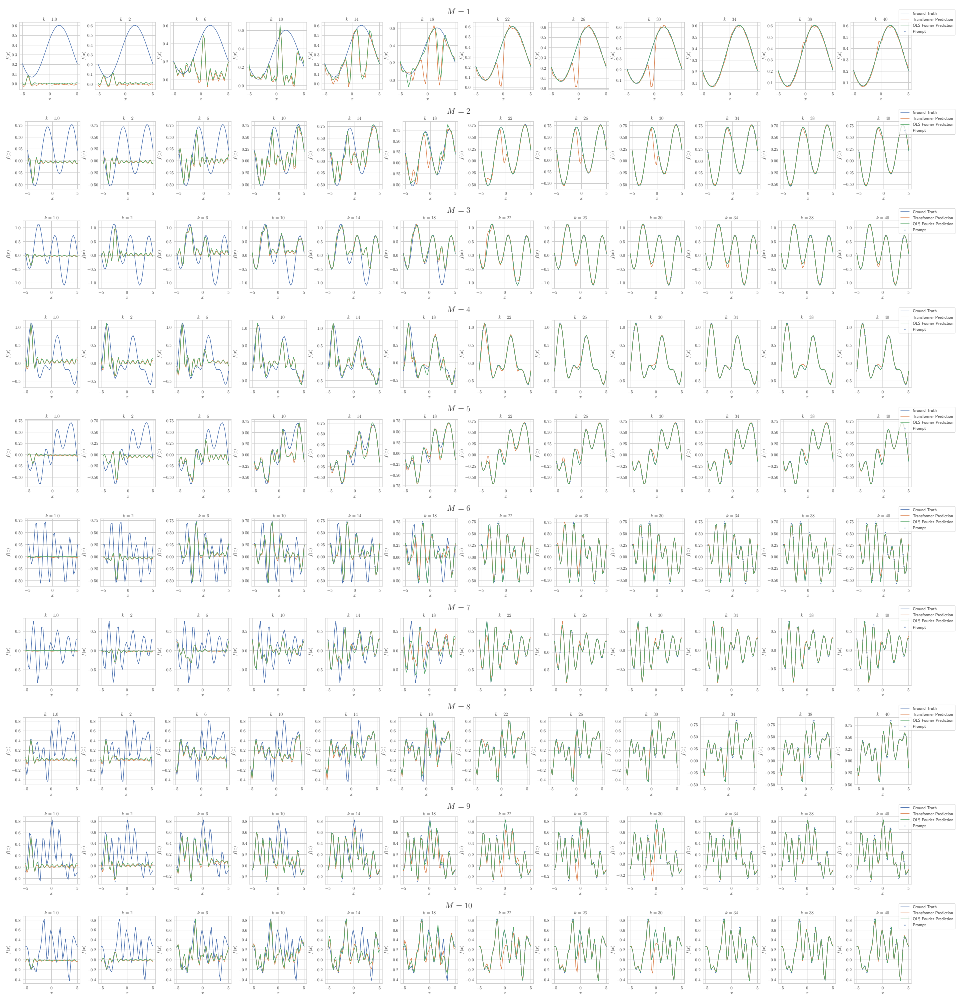
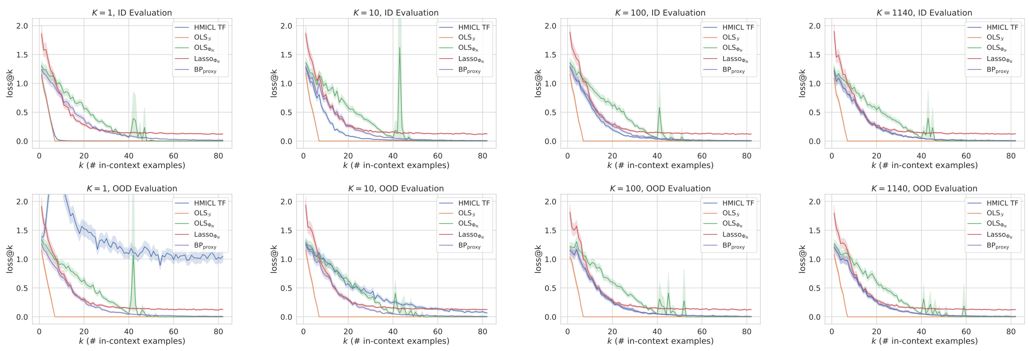


Figure 9: Visualizing the functions simulated by the transformer and the OLS Fourier Basis, for different values of M (maximum frequency) and k (number of in-context examples)

Table 2: **Multi-task generalization results for Fourier Series problem**. The first row is ID evaluation, second row is OOD evaluation. As task diversity (K) increases, the model starts behaving like Lasso$_{\Phi_N}$ and BP$_{\text{proxy}}$, and its ID and OOD losses become almost identical, i.e. it generalizes to OOD.



C Details regarding Multi-task generalization experiments

C.1 Monomials Multi-task

Plots for various task diversities that we experiment with are shown in Table 1.

C.2 Fourier Series Multi-task

Fourier Series problem – MICL setting. Please refer to the setup defined in §B.2.2, which comes under the MICL setting as it corresponds to a single function class, $\mathcal{F}^{\text{fourier}}_{\Phi_N}$.

Extending Fourier Series problem to HMICL setting. For extension to HMICL, we use multiple subsets of frequencies $\mathcal{S}_k$'s to define the mixture. Each $\mathcal{S}_k$ defines a function class $\mathcal{F}^{\text{fourier}}_{\mathcal{S}_k}$. The pretraining distribution is induced by the uniform distribution $\mathcal{U}(\boldsymbol{\mathcal{F}})$ over a collection of such function classes, $\boldsymbol{\mathcal{F}} = \{\mathcal{F}^{\text{fourier}}_{\mathcal{S}_1}, \cdots, \mathcal{F}^{\text{fourier}}_{\mathcal{S}_K}\}$, where $\mathcal{S}_k \subset \Phi_N(x)$, the full basis. For example, $\mathcal{S}_k$ could be $[1, \cos(2\pi x/L), \cos(6\pi x/L), \cos(9\pi x/L), \sin(2\pi x/L), \sin(6\pi x/L), \sin(9\pi x/L)]^T$, consisting of sine and cosine frequencies corresponding to integers $2, 6$ and 9. (Note that 1, the intercept term, is a part of every $\mathcal{S}_k$). K feature sets $\mathcal{S}_k$'s, each of size D, are chosen at the start of the training and remain fixed. K is the task diversity of the pretraining distribution. To sample a training function for the TF, we first sample a function class $\mathcal{F}^{\text{fourier}}_{\mathcal{S}_k}$ with replacement from $\mathcal{U}(\boldsymbol{\mathcal{F}})$ and then sample a function from the chosen class; $f(\boldsymbol{x}) = \boldsymbol{w}^T \mathcal{S}_k(\boldsymbol{x})$, where $\boldsymbol{w} \sim \mathcal{N}_D(\boldsymbol{0}, \boldsymbol{I})$. Similar to the Monomials problem, our aim is to check if TF trained on $\mathcal{U}(\boldsymbol{\mathcal{F}})$ can generalize to the full distribution of all function classes (for feature sets of size D) by evaluating its performance on function classes corresponding to feature sets $\mathcal{S}' \notin \{\mathcal{S}_1, \cdots, \mathcal{S}_K\}$.

Training Setup. $d = 1, p = 82, N = 20$. So, the full basis, $\Phi_N(x)$, had 20 frequencies. $D = 3$. We experiment with $K \in \{1, 10, 100, 200, 400, 800, 1140\}$.

Evaluation Setup. The baselines we consider are $\text{OLS}_{\mathcal{S}}$, OLS_{Φ_N}, and Lasso_{Φ_N} For Lasso, we use $\alpha = 0.1$. We again evaluate in two settings: **(a) ID**: On functions from the pretraining distribution. **(b) OOD**: On functions not in the pretraining distribution by sampling from a function family not corresponding to any of the $\mathcal{S}_k's$ used to define the pretraining distribution.

Results. Plots for various task diversities that we experiment with are in Table 2. The trend is the same as it was for Monomials problem, i.e. ID performance degrades and OOD performance improves as K increases. As K increases, TF's performance on ID and OOD becomes identical (from $K = 100$ onwards) and similar to the Lasso_{Φ_N} and BP_{proxy} baselines.

C.3 Details on the phenomenon of forgetting

Problem Setup. We follow the Noisy Linear Regression (NLR) setup from Raventós et al. [2023]: $d = 8, p = 15$ (without curriculum learning). The noise variance $\sigma^2 = 0.25$. For this problem, the transformer has 8 layers, 128-dimensional embeddings, and 2 attention heads, and is trained with a batch size of 256 for 500k steps. One-cycle triangle learning rate schedule Smith and Topin [2018] is used with 50% warmup. Detailed plots for the four groups of task diversities mentioned in §4 are in Figure 10. OOD loss curves with Bayesian predictors for various checkpoints for task div 2^8 are in Figure 11. For other representative task diversities, the OOD loss curves of TF and Bayesian predictors are in Table 3. Plots showing mean squared errors of implied weights of TF (found as per §B.1) with Bayesian predictors are in Table 4.

Classification of task diversities. ID loss ≈ 0 for all task diversities during pretraining. We group them into the following 4 categories based on OOD loss:

1. 2^1 to 2^3 (no generalization; no forgetting) – OOD loss never decreases, converges to a value worse than or same as at the start of the training (t_0), agrees with dMMSE at the end of the training (t_{end}). [Figure 10a]

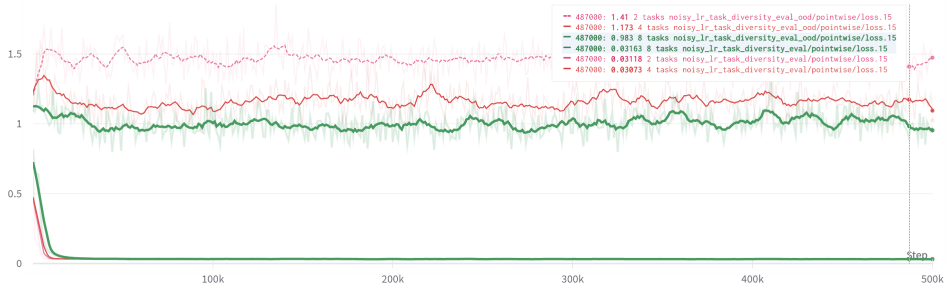
2. $\mathbf{2^4}$ to $\mathbf{2^6}$ (some generalization and forgetting) – OOD loss improves, reaches a minima t_{min}, then worsens. OOD loss is worse than ID loss throughout pretraining and agrees with dMMSE at t_{end} (i.e., any generalization to Gaussian distribution is forgotten by t_{end}). [Figure 10b]

3. $\mathbf{2^7}$ to $\mathbf{2^{11}}$ (full generalization and forgetting) – OOD loss improves, reaches a minima t_{min}, at which it is same as ID loss, then it worsens. At t_{min}, OOD loss agrees with Ridge (Figure 11a), then gradually deviates from it and at t_{end}, it is in between dMMSE and Ridge (e.g., Figure 11c). We refer to this group of task diversities as the "Gaussian forgetting region" since **the model generalizes to the full (Gaussian) distribution over tasks at $\mathbf{t_{min}}$ but forgets it by $\mathbf{t_{end}}$**. [Figures 10c, 11]

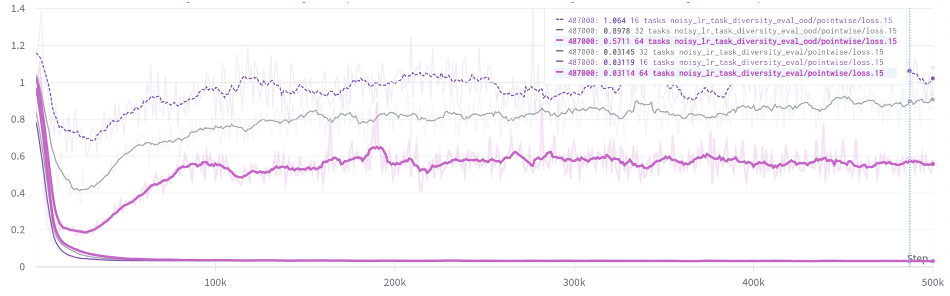
4. $\mathbf{2^{12}}$ to $\mathbf{2^{20}}$ (full generalization; no forgetting) – Throughout pretraining, OOD and ID losses are identical and OOD loss agrees with Ridge. [Figure 10d]

Relation to Simplicity bias? The phenomenon of forgetting (displayed by task diversity groups 2 and 3 above) is an interesting contrast to the grokking literature and in particular to Nanda et al. [2023], where they find that the model first memorizes and then generalizes (which, on the surface, is the opposite of what we observe). We can explain forgetting from the perspective of simplicity bias. Since PT$_{\text{dist.}}$ is discrete and perhaps contains lots of unnecessary details, the model instead finds it easier to generalize to the 'simpler' Gaussian distribution which is continuous and much more nicely behaved. Hence, **we speculate that the simplicity of the PT$_{\text{dist.}}$ is inversely proportional to the number of tasks it contains**. Very small task diversities (group 1) are exceptions to this rule since their PT$_{\text{dist.}}$ is arguably much simpler than FG$_{\text{dist.}}$. So, we do not see forgetting in those cases as the model prefers to only learn PT$_{\text{dist.}}$. Thus, we hypothesize that the simplicities of the distributions have the following order (G_i denotes group i): PT$_{\text{dist.}}(G_2) \approx$ PT$_{\text{dist.}}(G_3) <$ PT$_{\text{dist.}}(G_4) <$ FG$_{\text{dist.}} \ll$ PT$_{\text{dist.}}(G_1)$.

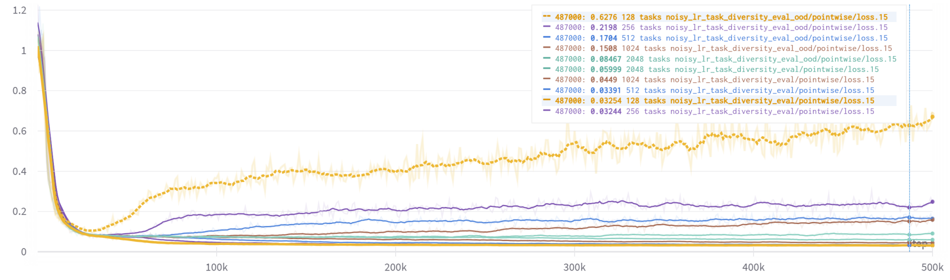
Robustness and effect of the number of dimensions. The phenomenon of forgetting is robust to changes in learning rate and its schedule (Figure 12), and to model sizes and position encodings (Monomials and Fourier Series multi-task setups use a 12-layer transformer that does not have position encodings). We also experimented with NLR problems having dimensions $d = 3$ and $d = 16$ (Figure 13) and found that the extent of forgetting (denoted by the disagreement between ID and OOD losses) is directly proportional to the input dimension (d). Note that following Raventós et al. [2023] we keep the signal-to-noise ratio (d/σ^2) constant across these experiments by adjusting the noise scale to ensure that noise has a proportional effect and the observations are due to change in dimension alone.



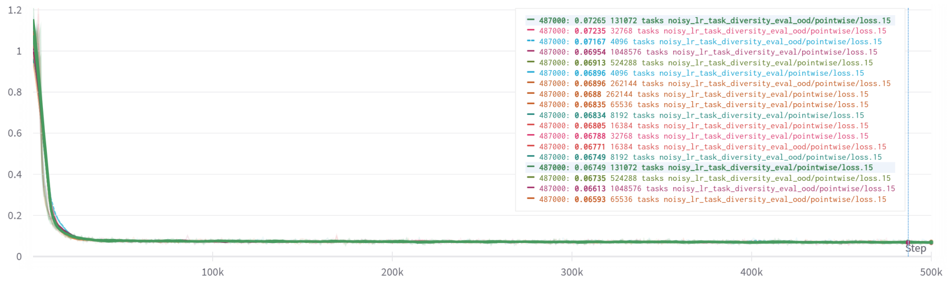
(a) No generalization; no forgetting)



(b) Some generalization and forgetting



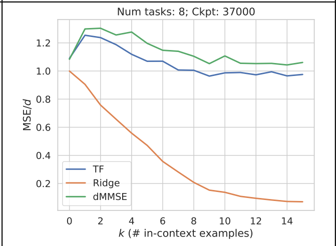
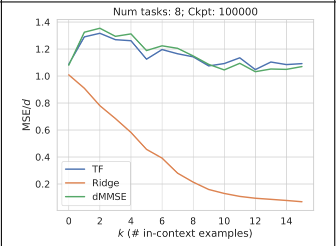
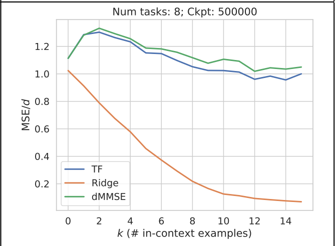
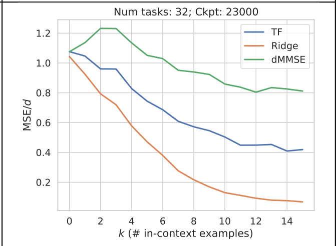
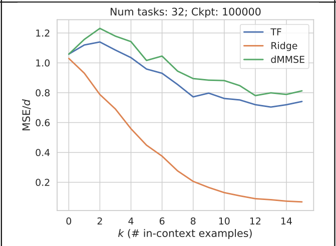
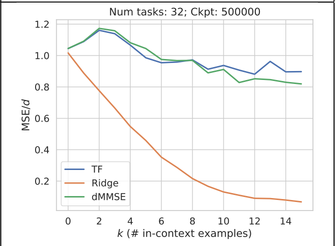
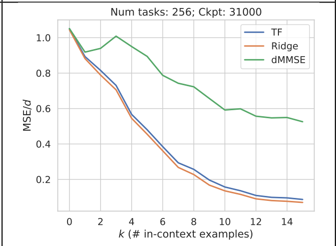
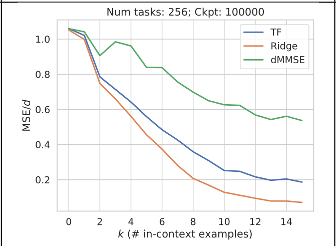
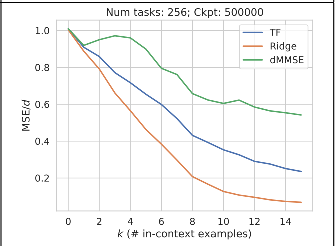
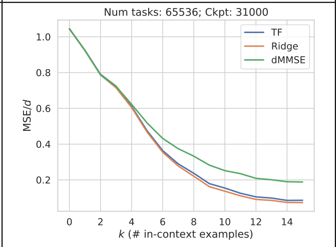
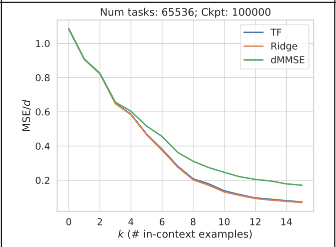
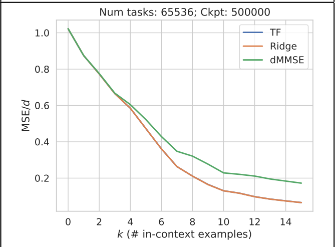
(c) Full generalization and forgetting

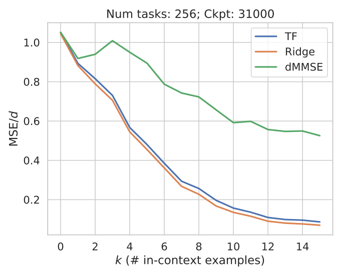


(d) Full generalization; no forgetting

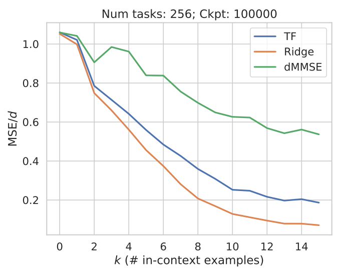
Figure 10: **Evolution of ID and OOD losses during pretraining for different task diversity groups for the Noisy Linear Regression problem. The forgetting phenomenon is depicted by groups in Figures (b) and (c)**. The moving average (over 10 train steps) of **ID (*eval)** and **OOD (*eval_ood)** losses are plotted, with the original (non-averaged) curves shown in a lighter shade. A checkpoint towards the end of the training is highlighted. We see that as we increase task diversity (i.e. go from group (a) towards (d)), the difference between ID and OOD losses decreases. Groups (b) and (c) are noteworthy as they display the phenomenon of forgetting, where the models' OOD loss at an earlier checkpoint is the same as ID loss, but it increases later.

Table 3: OOD loss curves of TF and Bayesian predictors for various checkpoints of models corresponding to task diversities (K) 2^3, 2^5, 2^8, 2^{16} respectively in rows. Each plot presents the loss across different prompt lengths. For task diversities 2^5 and 2^8, plots in the first column represent the point of minima (t_{min}). For task diversities 2^3 and 2^{16}, plots in the first column represent an earlier checkpoint.

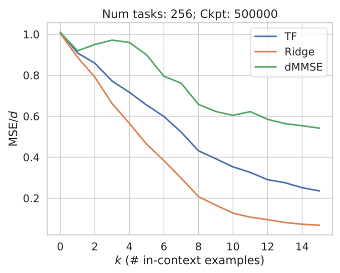
K	minima (t_{min}) or an earlier checkpoint	checkpoint after 100k train steps	checkpoint after 500k train steps
2^3			
2^5			
2^8			
2^{16}			



(a) after 31k train steps (t_{min})



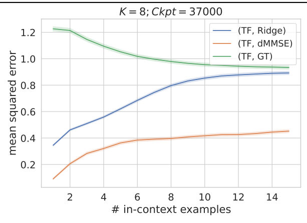
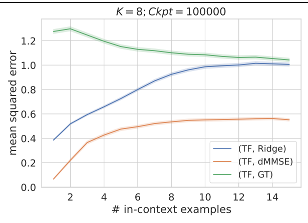
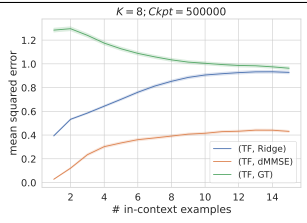
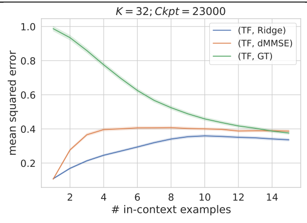
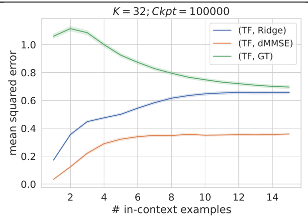
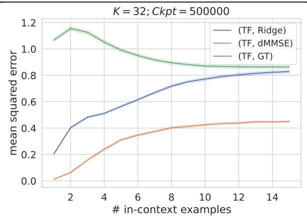
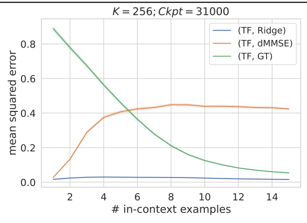
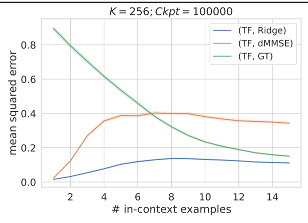
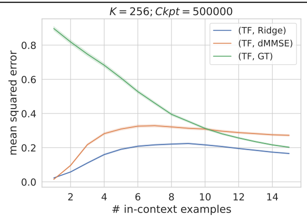
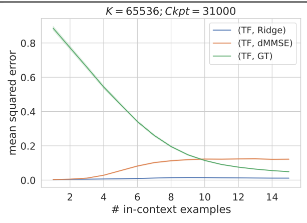
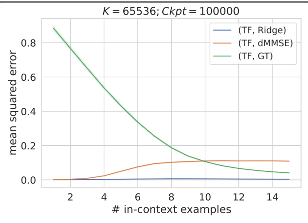
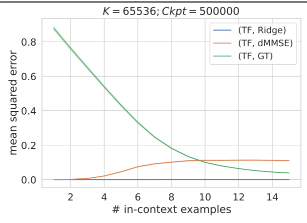
(b) after 100k train steps



(c) after 500k train steps

Figure 11: Plotting the OOD loss for various checkpoints during training for task diversity 2^8, along with the Bayesian predictors. At t_{min}, the model agrees with Ridge regression for all prompt lengths but later deviates and converges to somewhere in the middle of two Bayesian predictors.

Table 4: Mean squared errors of implied weights of TF with Bayesian predictors during OOD evaluation for various checkpoints of models corresponding to task diversities (K) 2^3, 2^5, 2^8, 2^{16} respectively in rows. Each plot presents the implied difference of weights across different prompt lengths. For task diversities 2^5 and 2^8, plots in the first column represent the point of minima (t_{min}). For task diversities 2^3 and 2^{16}, plots in the first column represent an earlier checkpoint.

K	minima (t_{min}) or an earlier checkpoint	checkpoint after 100k train steps	checkpoint after 500k train steps
2^3			
2^5			
2^8			
2^{16}			

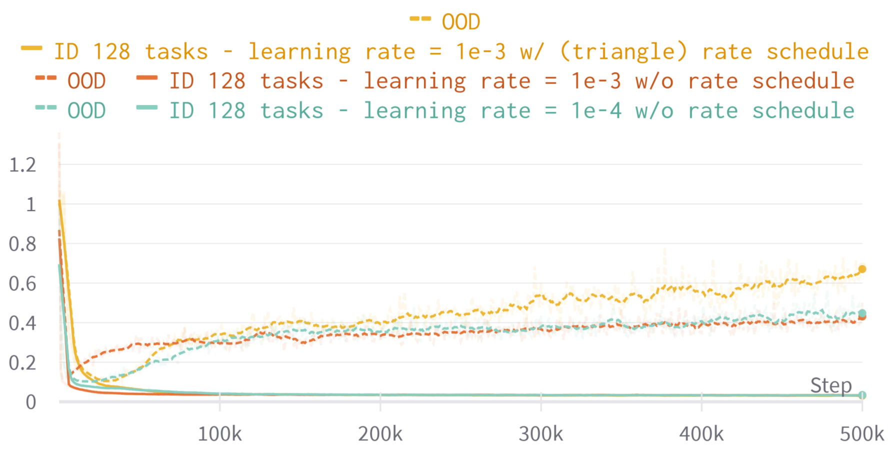


Figure 12: The moving average (over 10 train steps) of ID (solid lines) and OOD (dashed lines) losses for task diversity 2^7 for different learning rates, with & without learning rate schedule are plotted. While the nature and extent of forgetting changes, the phenomenon itself is robust and is observed across all settings.

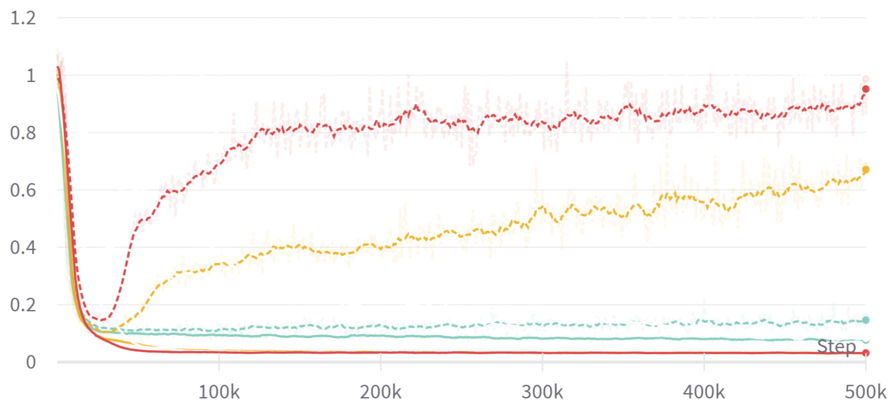


Figure 13: The moving average (over 10 train steps) of ID (solid lines) and OOD (dashed lines) losses for task diversity 2^7 for different input dimensions (3 (green), 8 (yellow), 16 (red)) are plotted. The extent of forgetting is directly proportional to the input dimension (d).