Can Transformers In-Context Learn Task Mixtures?

Nilesh Tripuraneni*, **Lyric Doshi***, **Steve Yadlowsky***
{yadlowsky, lyric, nileshtrip}@google.com
Google DeepMind

Abstract

In-context learning (ICL) refers to the ability of Large Language Models (LLMs) to perform new tasks by conditioning on input-output samples without any parameter updates. Previous work has established that, in a controlled setting, transformers can optimally perform ICL for tasks from a single task family, here a single function class, when they are pretrained on example tasks from that family. Using this setting, we probe the relationship between the pretraining data mixtures and downstream ICL performance. In particular, we empirically explore the ability of pretrained transformers to *select a family of tasks* (i.e. amongst distinct function classes) and *perform learning within that task family* (i.e. learn a function within a function class), all in-context. We show, for pretraining task mixtures balanced across task families, the cost of unsupervised downstream ICL task-family selection is near-zero. For task families rarely seen in pretraining, downstream ICL learning curves exhibit complex, task-dependent non-monotonic behavior. We also characterize the benefit of conditional pretraining in this simplified model, showing how task-family instructions can reduce the overhead of in-context task-family selection.

1 Introduction

Large Language Models (LLMs) can perform in-context learning (ICL) – condition on a prompt sequence consisting of "in-context" samples to perform new tasks without explicit model training on those tasks [Brown et al., 2020]. A line of work seeks to better understand this ability through a mixture of empirical and theoretical exploration [Xie et al., 2022, Min et al., 2022, Olsson et al., 2022]. In this work, we characterize the interaction between the pretraining task composition and the downstream sample complexity of ICL. We build on the work of Garg et al. [2022], which proposes a controllable and tractable setting for probing the behavior ICL in transformer models. Garg et al. [2022] pretrains decoder-only transformers on prompts of the form $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \ldots$ for $\mathbf{x}_i$ sampled from a data distribution over $\mathbb{R}^d$ and f sampled from a real-valued function class (e.g. linear models). Empirically, Garg et al. [2022] show transformers can perform ICL on unseen functions (i.e. a task) from the same function class as seen in pretraining, with performance close to the optimal standard machine learning estimator for that task. For example Garg et al. [2022] observes transformers can match the performance of the least squares estimator when given data from linear models, the Lasso when given data from sparse linear models, and ReLU networks when given data from ReLU networks. Later work has investigated the mechanistic underpinnings of how transformers may implement ICL [Akyürek et al., 2023, Dai et al., 2023] (i.e. via gradient descent) and their generalization properties [Abernethy et al., 2023, Li et al., 2023, Bai et al., 2023].

In our work, we pretrain on mixtures of *task families*, each of which is a function class such as linear models or ReLU networks. Our goal is low prediction error on a test point from a specific *task* f selected from a task family. We ask: *How many samples are required for a transformer to both select*

*Equal contribution.

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

a task family and perform in-context learning on unseen tasks from that task family in-context?[2] *Can transformers pretrain for tasks from multiple task families without exhibiting task interference?*. Most prior work focuses on pretrained transformers learning *single* function classes, i.e. a single task family. Understanding the performance on ICL each downstream task family embodies a form of distribution shift: the model is shown multiple different task families in pretraining, but must predict at test-time using ICL on a prompt sequence from each single task family. Some task families may have seen rarely in pretraining. The model must both in-context identify the specific task family and in-context learn an unseen task within that family. A more detailed description of related work is provided in Appendix A.1. We make the following observations and contributions:

- We pretrain transformers on prompts on weighted task mixtures of different function classes. We empirically gauge their downstream ICL learning curves (or sample complexity) on their individual constituent task families. We observe that the cost of unsupervised task-family selection is almost zero for evenly-weighted pre-training mixtures relative to the baseline optimal sample complexity of models that have only seen that single task family in pretraining (Fig. 1a for example). That is, these pretrained models do not suffer from task interference [Wang et al., 2019, Zhao et al., 2018].
- When pretrained on skewed task mixtures (with one function class rarely seen), the downstream ICL sample complexity increases on the rarely-seen tasks. The phenomenology of the learning curves is nuanced and task-dependent: we actually observe non-monotonic, sample-wise double descent-like behavior [Nakkiran et al., 2021, Hastie et al., 2022] in some cases as the estimator error increases then decreases with the number of ICL samples (Fig. 1a and Fig. 4a). In others, the task-family selection cost remains a constant overhead even with many in-context samples (Fig. 3a).
- We investigate the benefits of conditional pretraining (i.e. providing in-context task keys as task-family-identifying instructions in both pretraining and at test-time) to circumvent the overhead of task-family selection. We observe task-conditioning often removes the cost of task-family selection (Fig. 1b and Fig. 4b). Instructions also asymmetrically help some tasks more than others.

2 Preliminaries and Set-up

We consider a data-generating model where d-dimensional covariates are drawn $\mathbf{x}_i \sim \mathcal{D}_{\mathcal{X}}$ and a (random) function $f \sim \mathcal{D}_{\mathcal{F}}$ is sampled. Like Garg et al. [2022], Akyürek et al. [2023], we frame the ICL problem as providing a single prompt sequence $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \ldots \mathbf{x}_n, f(\mathbf{x}_n), \mathbf{x}_{n+1}$ to the model (i.e. a transformer) and generating a prediction for $f(\mathbf{x}_{n+1})$: $\tilde{f}(\mathbf{x}_{n+1})$. For example, for a linear data-generating model, a single f can be generated by drawing $\boldsymbol{\beta} \sim \mathcal{N}(0, \mathbf{I}_d)$ and defining $f(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x}$. The performance of an in-context learner is judged by its predictive squared-loss $\mathbb{E}[(\tilde{f}(\mathbf{x}_{n+1}) - f(\mathbf{x}_{n+1}))^2]$, with the expectation taken over the randomness in the prompt and query.

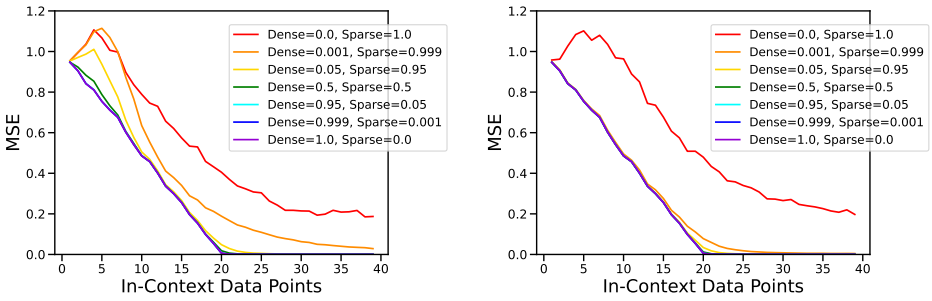
Following Garg et al. [2022]'s approach to construct a sequence of n (d-dimensional $\mathbf{x}$, 1-dimensional $f(\mathbf{x})$) values, we pad $f(\mathbf{x})$ with 0s to make it d-dimensional and produce a sequence of $2n$ real-valued vectors, or "tokens", alternating between $\mathbf{x}_i$ and $f(\mathbf{x}_i)$. We insert one (learnable) linear layer to project each input "token" into the transformer's model dimension, and we insert another to reverse the embedding and produce a d-dimensional output representing the next $\mathbf{x}$ or $f(\mathbf{x})$, as per sequence position. The training-time loss uses the next-token prediction squared loss computed only over the $\mathbf{x}_i$ positions, which predict $\tilde{f}(\mathbf{x}_{n+1})$, and ignores the $f(\mathbf{x}_i)$ positions, which nominally predict $\tilde{\mathbf{x}}_{i+1}$. As in prior work, the model is pretrained *tabula rasa* on simulated data for different data-generating setups. We do not fine-tune a pretrained language model and do not train on actual text.

We evaluate a GPT-2 scale decoder-only transformer trained on data sampled from mixtures of pairs of function classes: {dense linear functions, sparse linear functions} and {dense linear functions, two-layer ReLU neural networks}. Like Garg et al. [2022], Akyürek et al. [2023], we use a data generator per function class to sample from $f \sim \mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$, $f \sim \mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$, and $f \sim \mathcal{D}_{\mathcal{F}_{\text{RELU}}}$. Mixture distributions then take the form: $\mathcal{D}_{\mathcal{F}} = w \cdot \mathcal{D}_{\mathcal{F}_1} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_2}$ for moderate and extreme values of w. See Appendices A.2 and A.3 for architectures, training parameters, and data generation details.

Concretely, each training batch consists of k prompts. Each prompt is a sequence of $\mathbf{x}_i, f(\mathbf{x}_i)$ pairs, with the $\mathbf{x}$'s sampled from $\mathcal{D}_{\mathcal{X}}$ (i.e. $\mathcal{N}(0, \mathbf{I}_d)$) and a single $f \sim \mathcal{D}_{\mathcal{F}}$ per prompt. We use a fixed w per

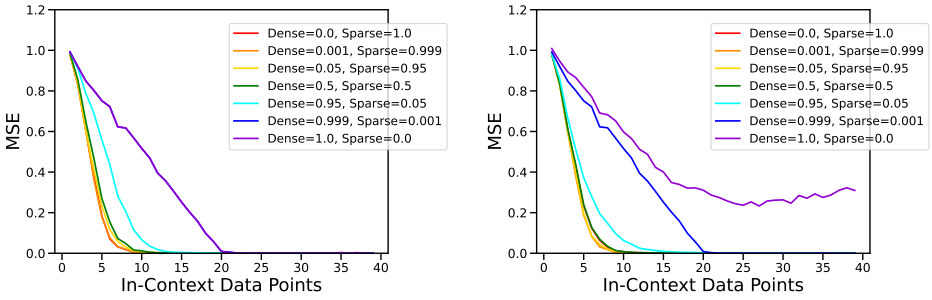
[2]This is the joint sample complexity of both model selection (i.e. selecting a task family in-context) and generalization to (learning within the task family in-context).

Figure 1: ICL learning curves for evaluations on prompts drawn $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Transformers were pretrained on mixtures of $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$. Different curves correspond to different w.



(a) Without Instructions (b) With Instructions

Figure 2: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ with (left) and without instructions (right). Transformers were pretrained on mixtures of $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$. Different curves correspond to different w.



(a) Without Instructions (b) With Instructions

pretraining run. We run each task mixture experiment with and without an task-family-identifying instruction encoded as a "special token" prepended to the prompt, as described in Appendix A.4. This bears resemblance to the conditional pretraining [Keskar et al., 2019, Korbak et al., 2023] and instruction tuning paradigms [Wei et al., 2022].

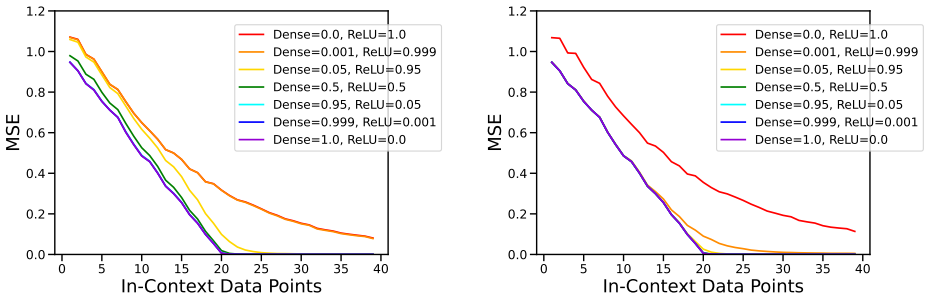
3 Results

To gauge the impact of the pretraining mixture on the downstream ICL performance, we evaluate their ICL learning curves. We input prompts with varying lengths of in-context samples and evaluate the averaged squared loss on their prediction for the next sample. Garg et al. [2022], Akyürek et al. [2023] show that transformers pretrained on a single task family have ICL learning curves matching the optimal sample complexity for the task's standard ML estimator so we do not re-plot those. For each evaluation, we sample batches for a single constituent function class at a time from the training task mixture distribution, including instructions when pretraining also included instructions.

First we evaluate the performance of transformers pretrained on weighted mixtures of $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ and $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$. Fig. 1a shows their performance evaluated on in-context examples from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$. For pretraining mixtures with sufficient data from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ (i.e. mixture weight $w \geq 0.5$), the cost relative to the baseline of being only pretrained on $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ data is negligible. For more extreme mixture weights, where the $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ data examples are rarely seen in pretraining, the cost of task-family selection is more evident as they do not match the baseline on training solely on data from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$. The ICL learning curves (colored gold, orange, red in Fig. 1a) display non-monotonic behavior where the error increases initially with more in-context data before decreasing[3]. In most cases, the cost of task-family selection in ℓ_2 error is a constant upfront cost for small numbers of data points that quickly decreases to the baseline. In Fig. 1b we see for all $w > 0$, instructions almost entirely remove the cost of task-family selection relative to the baseline of only being pretrained on data from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$.

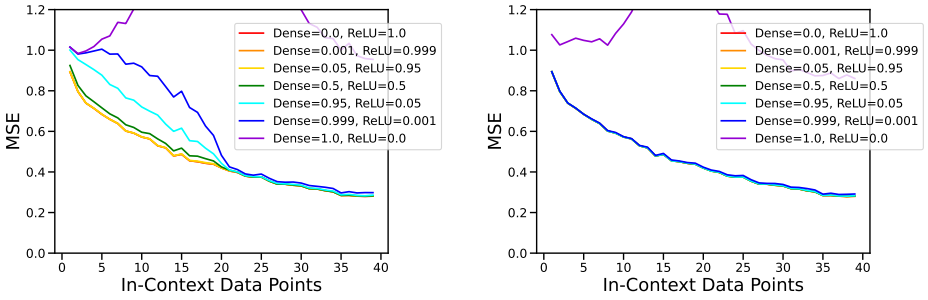
[3]This is reminiscent of sample-wise double descent behavior [Nakkiran et al., 2021, Hastie et al., 2022].

Figure 3: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Transformers were pretrained on mixtures of $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{RELU}}}$. Different curves correspond to different w.



(a) Without Instructions (b) With Instructions

Figure 4: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ with (left) and without instructions (right). Transformers were pretrained on mixtures of $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{RELU}}}$. Different curves correspond to different w.



(a) Without Instructions (b) With Instructions

We evaluate the same models on in-context data from $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ in Fig. 2a left. For evenly weighted pretraining mixtures with a sufficient fraction of $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ data prompts in pretraining, the cost of task-family selection relative to the baseline is small. For cases where $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ data is rarely seen, there appears to be a constant cost and performance looks akin to evaluation on $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$. In Fig. 2b, the instructions generally help but only slightly reduce the cost of task-family selection in some cases (i.e. the teal and blue curves). Adding task-family-identifying instructions when pretraining on $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ data and evaluating $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ data results in worse performance than not having instructions as the model is provided a task instruction value for $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ it never saw in pretraining.

Fig. 3 and Fig. 4 show similar evaluations with models pretrainined on $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ and $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$. We confirm for evenly weighted task mixtures, the pretrained transformer obtains near-optimal sample complexity curves with respect to the individual constituent in-context task family, while extreme task mixtures sacrifice sample complexity on the rarely seen task. In Fig. 3a left, the task-family selection cost appears to be constant throughout the learning curve (i.e. the gold curve) as opposed to a larger upfront cost that decays in Fig. 1a. We believe this may be because $\mathcal{F}_{\text{DENSE}}$ and $\mathcal{F}_{\text{RELU}}$ are not included in each other. The learning curves for extreme mixture weights also exhibit a non-trivial change in curvature away from the baseline $w = 0$ in Fig. 4a. We see the significant benefit of in-context instructions in Fig. 3b and Fig. 4b, again with nuanced task-dependence. Aside from the expected trouble with the $w = 1$ case, the instructions in Fig. 4b result in near perfect alignment with pretraining only on data from $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ ($w = 0$) when pretrained with few ReLU-generated examples (w close to 1). The learning curves at extreme mixture weights invert their curvature between Fig. 4a and Fig. 4b. Instructions also help in Fig. 3b (the orange curve) but are not enough to achieve the optimal baseline sample complexity. In Appendix A.5 we conduct experiments varying the transformer model size and show instructions are only beneficial at sufficiently large model sizes.

4 Conclusion

We empirically explored the interaction between the pretraining data composition and the downstream ICL performance on different task families. Gaining more theoretical understanding of the ICL task-family selection cost and non-trivial phenomenology in the learning curves is a direction for further research. Another direction is extending these results to a controlled natural-language setting.

References

J. Abernethy, A. Agarwal, T. V. Marinov, and M. K. Warmuth. A mechanism for sample-efficient in-context learning for sparse retrieval tasks, 2023.

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

Y. Bai, F. Chen, H. Wang, C. Xiong, and S. Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection, 2023.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, and F. Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers, 2023.

S. Garg, D. Tsipras, P. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=flNZJ2eOet.

T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949 – 986, 2022. doi: 10.1214/21-AOS2133. URL https://doi.org/10.1214/21-AOS2133.

N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

T. Korbak, K. Shi, A. Chen, R. V. Bhalerao, C. Buckley, J. Phang, S. R. Bowman, and E. Perez. Pretraining language models with human preferences. In *International Conference on Machine Learning*, pages 17506–17533. PMLR, 2023.

Y. Li, M. E. Ildiz, D. Papailiopoulos, and S. Oymak. Transformers as algorithms: Generalization and stability in in-context learning, 2023.

S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (12):124003, 2021.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. In-context learning and induction heads, 2022.

A. Raventós, M. Paul, F. Chen, and S. Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Z. Wang, Z. Dai, B. Póczos, and J. Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11293–11302, 2019.

J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners, 2022.

S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=RdJVFCHjUMI`.

X. Zhao, H. Li, X. Shen, X. Liang, and Y. Wu. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

A Appendix

A.1 Related Work

Raventós et al. [2023] investigates the role of pretraining function diversity for in-context learning in the setting of pure linear regressions–arguing that a sufficiently diverse distribution over linear tasks in pretraining is needed for ICL at test-time.

Closest to our work is that of Bai et al. [2023] which also explores transformers abilities to perform task-family selection on theoretical and empirical grounds, in addition to providing rigorous theoretical guarantees for transformers generalization properties in pretraining, their expressive power and their downstream in-context prediction performance. However the task-family selection guarantees and empirics in this paper are restricted to explorations within the same model family – namely studying task-family selection across evenly weighted task mixtures of linear regression with different label noise strengths and linear/logistic regression; that is selecting between different linear families. In contrast our work solely focuses on empirical investigations, but explores generalization across varying tasks as a function of their pretraining data composition to uncover phenomenology in downstream ICL curves.

A.2 Model Architecture

We trained a decoder-only Transformer [Vaswani et al., 2017] GPT2-sized model implemented in the Jax-based machine learning framework, Pax[4] with 12 layers, 8 attention heads, and a 256-dimensional embedding space (9.5M parameters) as our base configuration [Garg et al., 2022]. We too set the dimensions of $\mathbf{x}$ as 20 and used standard cosine positional embeddings in our model. We trained 1 million steps with a training batch size of 1024, using the Adam optimizer with standard hyperparameters set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-9}$ and no weight-decay. We used a linear ramp-up schedule followed by an inverse-square root learning rate decay. We empirically tuned to find a learning rate of 1 with 5,000 warm-up steps to be effective. Our results are presented without label noise in the data generation although find similar results adding label noise.

In evaluation, we used batch sizes of 8192.

A.3 Data Generation and Function Classes

Throughout our paper we always use $\mathcal{D}_{\mathcal{X}} = \mathcal{N}(0, \mathbf{I}_d)$ for the covariate distribution and consider models in dimension $d = 20$. For the function classes considered in the text we use:

- For $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ we generate a random $\boldsymbol{\beta} \sim \mathcal{N}(0, \mathbf{I}_d)$ and define $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}} = \{f(\mathbf{x}_i) : f(\mathbf{x}_i) = \boldsymbol{\beta}^\top \mathbf{x}_i/\sqrt{d}\}$.
- For $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ we consider sparse linear models with the number of non-zero elements set to $s = 2$. To generate the underlying parameter vector we randomly sample s coordinates uniformly without replacement from the $d = 20$ dimensional support and in each non-zero coordinate generate a random coefficient with distribution $\mathcal{N}(0, 1)$ to create $\boldsymbol{\beta}$. We then define $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}} = \{f(\mathbf{x}_i) : f(\mathbf{x}_i) = \boldsymbol{\beta}^\top \mathbf{x}_i/\sqrt{s}\}$.
- For $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ we consider two-layer ReLU networks with randomly generated weights. In particular we generate a first layer weight matrix $\mathbf{W} \in \mathbb{R}^{d_h \times d}$ where $d_h = 100$ is the hidden dimension and $\mathbf{W}_{ij} \sim \mathcal{N}(0, 2)$ and second layer coefficient $\boldsymbol{\beta} \sim \mathcal{N}(0, \mathbf{I}_{d_h})$. We then define the function class as $\mathcal{D}_{\mathcal{F}_{\text{RELU}}} = \{f(\mathbf{x}_i) : f(\mathbf{x}_i) = \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}_i)/(\sqrt{d \cdot d_h})\}$, where the activation function $\sigma(\cdot)$ is taken as the standard ReLU unit.

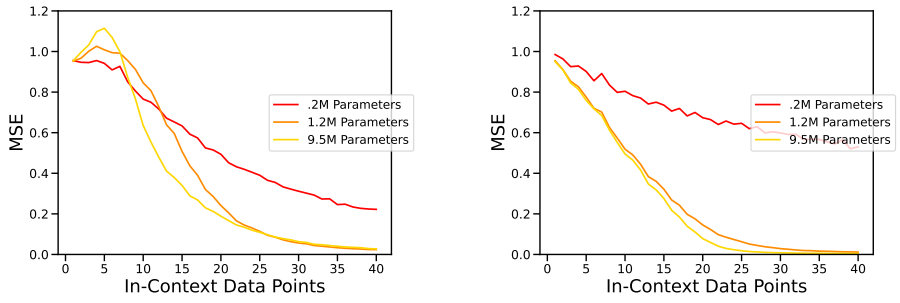
Note that in each case we define the normalization of each function class such that $\text{Var}(f(\mathbf{x}_i)) = 1$ to ensure the norms of the outputs are comparable across different function classes.

A.4 Instruction Encoding

We encode instructions as a special sample point prepended to each prompt sequence. The "$\mathbf{x}$" value of the point is a one-hot encoding of the task index. The $f(\mathbf{x})$ value is set to 0. With an instruction, a

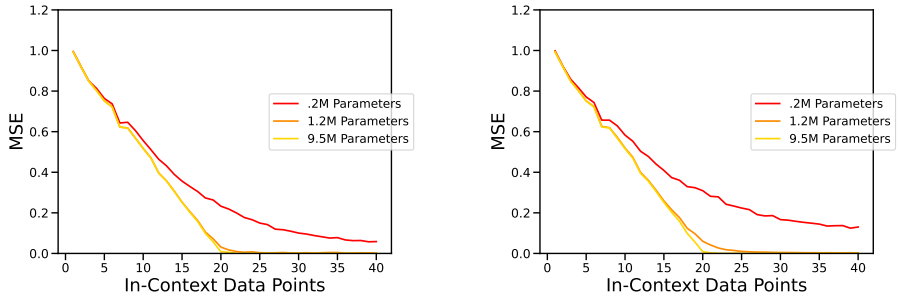
[4] `https://github.com/google/paxml`

Figure 5: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Curves reflect transformers of varying sizes pretrained on mixtures heavily favoring $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$, with $w = .001$ in $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$.



(a) Without Instructions (b) With Instructions

Figure 6: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ with (left) and without instructions (right). Curves reflect transformers of varying sizes pretrained on mixtures heavily favoring $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$, with $w = .999$ in $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$.



(a) Without Instructions (b) With Instructions

prompt takes the form ONE_HOT$(\mathcal{F}_{id}), \mathbf{0}, \mathbf{x}_1, f(\mathbf{x}_1), \ldots$, where $\mathcal{F}$ corresponds to the family from which f is sampled for this prompt and the $\mathcal{F}_{id}$ is a fixed index value representing that function class.

Empirical evidence suggests that the model both learns to attend to this special point when determining the function class (in so far as it does this explicitly internally) and learns to ignore it when solving for the function class parameter values.

A.5 Model Sizes Plots

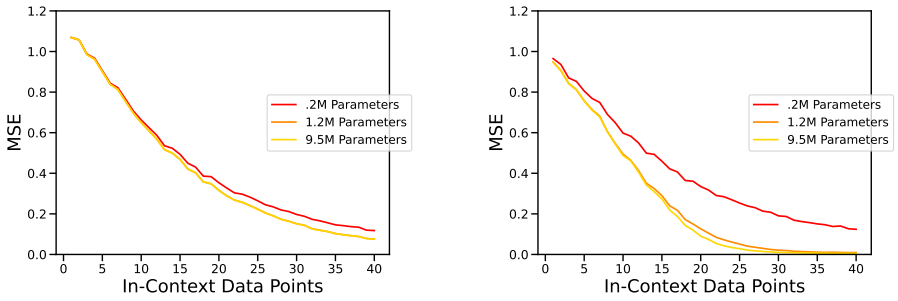
We explore the effects of different pretrained model sizes on downstream ICL performance in the case where we have extreme mixture weights. Overall we see two generic phenomena. As Figs. 5a, 6a, 7a and 8a show, the larger capacity models (with 1.2M and 9.5M parameters) generally perform better then the small .2M parameter model. The performance of the 1.2M and 9.5M models is roughly comparable. We also observe in Figs. 5b, 6b, 7b and 8b that Tiny model (.2M parameters) not only struggles to benefit from the instruction token/conditional pretraining procedure but usually performs worse. On the other hand, the higher capacity models (1.2M and 9.5M) have comparable performance to each other and benefit significantly from instruction tokens.

We used the same model sizes and architectures as Garg et al. [2022]'s Section A.1 with models Tiny, Small, and Standard.

A.6 Mixtures of Three Function Classes

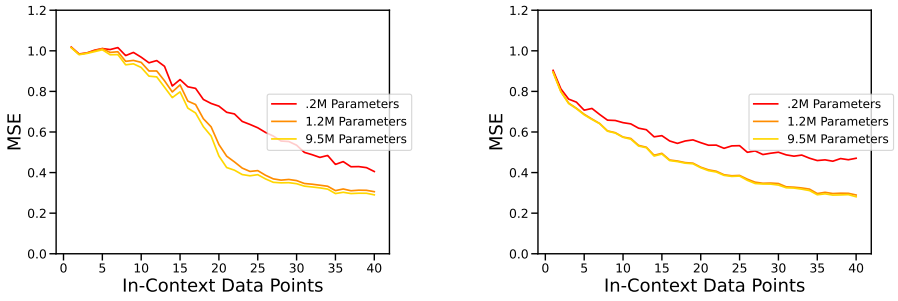
The experiments thus far used data mixtures drawn from two function classes. Here, we train a model drawing from three function classes and compare with the performance of the two-function-class case

Figure 7: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Curves reflect transformers of varying sizes pretrained on mixtures heavily favoring $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$, with $w = .001$ in $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{RELU}}}$.



(a) Without Instructions (b) With Instructions

Figure 8: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ with (left) and without instructions (right). Curves reflect transformers of varying sizes pretrained on mixtures heavily favoring $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$, with $w = .999$ in $w \cdot \mathcal{D}_{\mathcal{F}_{\text{DENSE}}} + (1-w) \cdot \mathcal{D}_{\mathcal{F}_{\text{RELU}}}$.

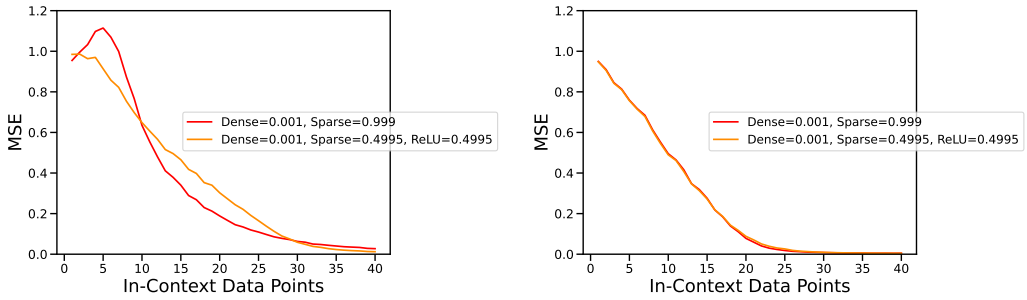


(a) Without Instructions (b) With Instructions

to see if the third class introduces greater task selection challenges. Each plot features the rarely-seen function class at the same rate in both the two- and three-function-class cases. Most notably, Fig. 9a shows that while the three-function-class curve avoids the initial larger bump in error, it's otherwise consistently worse by a constant amount until nearly 30 samples. We observed this constant error in Fig. 3a as well. In the case of Fig. 11a, we find the three-function-class mixture outperforms the two-function-class mixture. One hypothesis of why this occurs is that $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ is is included in $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ and hence may enable the model to learn the $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ task better as well.

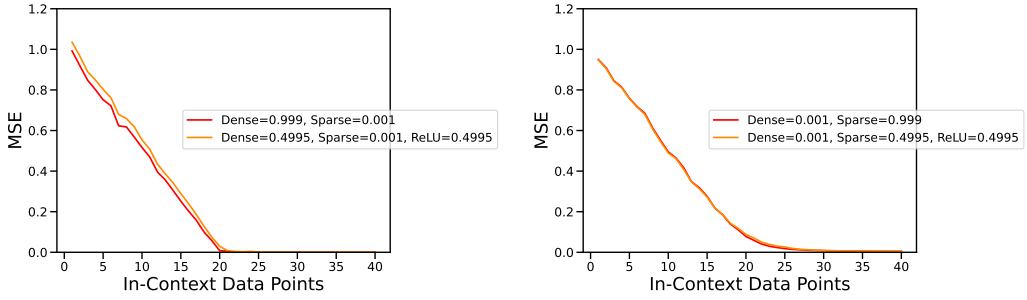
In every scenario, Figs. 9b, 10b, 11b and 12b show that using instructions results in the same behavior in between the two- and three-function-class cases.

Figure 9: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Curves reflect transformers pre-trained with varying number of function classes/tasks. In each case, $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ is equally rare, occurring at rate .001 while the remaining samples are split between the remaining classes.



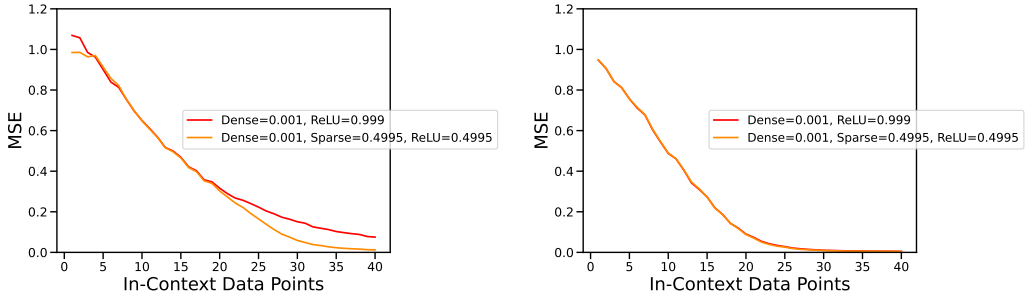
(a) Without Instructions (b) With Instructions

Figure 10: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ with (left) and without instructions (right). Curves reflect transformers pre-trained with varying number of function classes/tasks. In each case, $\mathcal{D}_{\mathcal{F}_{\text{SPARSE}}}$ is equally rare, occurring at rate .001 while the remaining samples are split between the remaining classes.



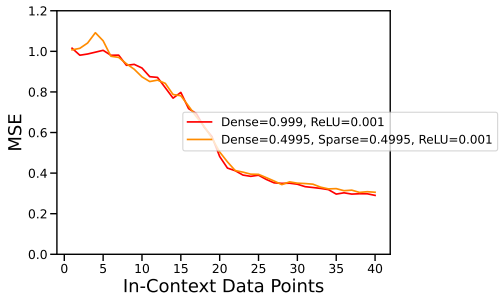
(a) Without Instructions (b) With Instructions

Figure 11: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ with (left) and without instructions (right). Curves reflect transformers pre-trained with varying number of function classes/tasks. In each case, $\mathcal{D}_{\mathcal{F}_{\text{DENSE}}}$ is equally rare, occurring at rate .001 while the remaining samples are split between the remaining classes.

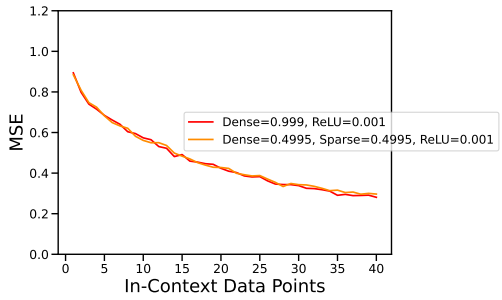


(a) Without Instructions (b) With Instructions

Figure 12: ICL learning curves for evaluations on prompts drawn from $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ with (left) and without instructions (right). Curves reflect transformers pre-trained with varying number of function classes/tasks. In each case, $\mathcal{D}_{\mathcal{F}_{\text{RELU}}}$ is equally rare, occurring at rate .001 while the remaining samples are split between the remaining classes.



(a) Without Instructions



(b) With Instructions