Can Transformer Models Generalize Via In-Context Learning Beyond Pretraining Data?

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

Abstract

Transformer models, notably large language models (LLMs), have the remarkable ability to perform in-context learning (ICL) – to perform new tasks when prompted with unseen input-output examples without any explicit model training. In this work, we study how effectively transformers can generalize beyond their pretraining data mixture to identify and in-context learn new tasks, i.e. functions, which are outside the pretraining distribution. The data mixtures are comprised of one or multiple *task families*, each of which is a function class such as linear models or sinusoids. To investigate this question in a controlled setting, we focus on the transformers ability to in-context learn functions from simulated data. While these models do well at generalizing to new tasks within the pretrained function class, we demonstrate various failure modes of transformers when presented with tasks which are out-of-distribution from their pretraining data. Together our results suggest that the impressive ICL abilities of high-capacity transformer models may be more closely tied to the coverage of their pretraining data mixtures than inductive biases that create fundamental generalization capabilities.

1 Introduction

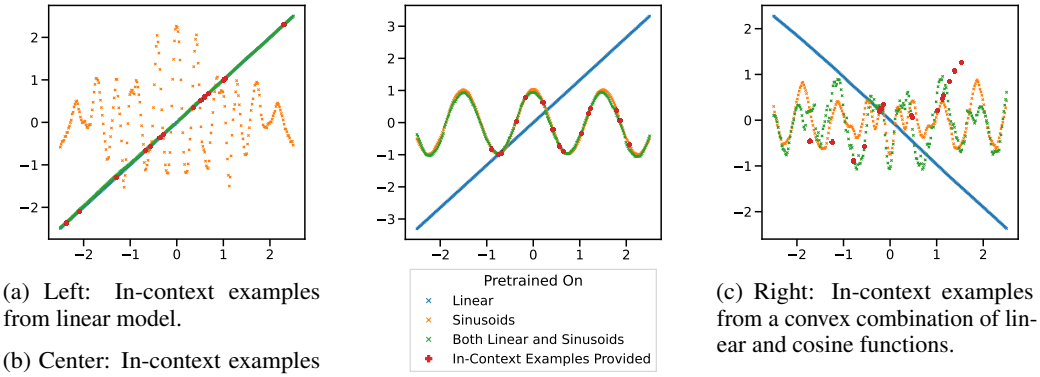
One of the impressive capabilities demonstrated by large language models is their ability to do few-shot learning by providing examples *in-context* and asking the model to generate a response to follow the final input provided [Brown et al., 2020]. Researchers have taken the underlying machine learning technology, transformer models, and demonstrated that they can perform also in-context learning tasks in domains other than language [Garg et al., 2022, Akyürek et al., 2022, Li et al., 2023]. In these works, they demonstrate the ability of transformers to learn high-dimensional and non-linear functions of the inputs from in-context examples, often matching or exceeding the performance of state-of-the-art machine learning models tuned for the purpose of learning those functions. For example, Garg et al. [2022] showed that after pretraining on sparse linear data, a transformer network can in-context learn unseen sparse linear functions as well as the Lasso, which is known to be statistically optimal for data being modeled.

In these models, as in the large language models that they are designed to reflect, pretraining (or fine-tuning) the model with relevant data to teach the model how to perform in-context learning is critical to enabling this capability. In this work, we focus in on a specific aspect of this pretraining process—the data used in pretraining—and investigate how it affects the few-shot learning capabilities of the resulting transformer model.

The few-shot learning setup that we study follows Garg et al. [2022], where the goal is to use a set of provided inputs and labels, $((\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots (\mathbf{x}_n, f(\mathbf{x}_n)))$ to make a prediction about the label $f(\mathbf{x}_{n+1})$ for a new input $\mathbf{x}_{n+1}$. The number of examples provided is small relative to the amount of data used to pretrain the meta-learning model used to perform this task (hence the "few-shot" nomenclature). A common approach to apply sequence models for few-shot learning is first to pass

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

Figure 1. Comparing predictions from models pretrained on different data sources after providing 3 sets of in-context examples (shown in red). Predictions are made by sweeping over $\mathbf{x}$ values passed in as the last element of the sequence after the in-context examples.



(a) Left: In-context examples from linear model.

(b) Center: In-context examples from cosine.

(c) Right: In-context examples from a convex combination of linear and cosine functions.

the examples in sequentially, alternating inputs and labels, as $(\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \ldots, \mathbf{x}_n, f(\mathbf{x}_n))$. Finally, the test input point $\mathbf{x}_{n+1}$ is passed as the final element of the sequence, and model prediction for the next item in the sequence is treated as the predicted label.

Given a data source $\mathcal{D}$ containing sequences $\mathbf{s}_i = (\mathbf{x}_{i,1}, f(\mathbf{x}_{i,1}), ..., \mathbf{x}_{i,n}, f(\mathbf{x}_{i,n}))$, we pretrain the parameters θ of the transformer model $m_\theta(\mathbf{s})$ by performing loss minimization on

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{s}_i \in \mathcal{D}} \sum_{j=1}^{n} \ell \left(f(\mathbf{x}_{i,j}), m_\theta(\mathbf{s}_{i,1:j}) \right), \tag{1.1}$$

for the squared-loss ℓ, where we use $\mathbf{s}_{i,1:j}$ to refer to the i-th sequence up to (but not including) the j-th output $f(\mathbf{x}_{i,j})$. We use a 9.5M parameter decoder model with 12 layers, 8 attention heads, and a 256-dimensional embedding space as in Garg et al. [2022]. Details on the model and our training setup are in Appendix A.1.

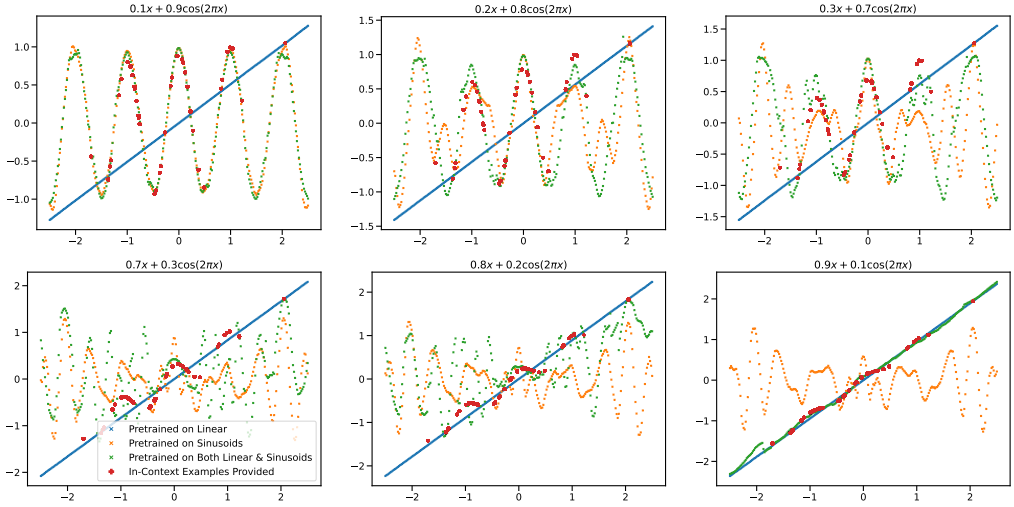
The focus of this paper is understanding how the construction of the data source $\mathcal{D}$ affects the in-context learning abilities of the model in the controlled setting of learning function classes. Garg et al. [2022], Akyürek et al. [2022], Von Oswald et al. [2023], Zhang et al. [2023] and Li et al. [2023] show that for certain function classes (such as linear functions and a few types of non-linear functions), when $\mathcal{D}$ is generated with functions all drawn from the same function class, $\mathcal{F}$, the model is able to perform in-context learning on new functions from this class. [Garg et al., 2022] argues transformers generalize well to tasks/function drawn from the same distribution as the training data.

However, one general open question is how these models perform on examples that are out-of-distribution from the training data. In the setting studied here, we interpret the generalization question as the following: *"Can a model generate good predictions with in-context examples from a function (task) not in any of the base function classes (task families) seen in the pretraining data mixture?"* In this paper, we turn this question into a series of precise empirical experiments, for which we find little evidence of good generalization.

2 Out-of-Distribution Generalization for In-Context Learning

We examine the ICL generalization capabilities of models along two axes. First, we motivate exploring and test ICL performance on functions the model has both never seen in training and also could plausibly predict: convex combinations of functions drawn from the pretraining function classes. Second, we evaluate ICL performance on functions which are extreme versions of functions seen in pretraining (ie., sinusoids with much higher or lower frequencies than those typically seen in pretraining). In both cases, we find little evidence of out-of-distribution generalization. When the function is significantly far from those seen during pretraining, the predictions are erratic. However, when the function is sufficiently close to the pretraining data, the model approximates it well with predictions from the function classes on which it was pretrained.

Figure 2. Predictions from transformer models pretrained on linear functions (blue), sinusoids (orange), or both (green) when provided the red examples in-context, coming from a combination of a linear function and sinusoid, with the relative weights noted in the titles of each plot.



Garg et al. [2022] showed that transformer models can effectively learn linear models–i.e., linear combinations of the inputs. Given that the models have the capability to combine the raw inputs, they may be able to combine a few predictions that they are capable of making into one. Hence, one may hypothesize that the model has some inductive bias that allows it to combine pretrained function classes in nontrivial ways. For instance, one may suspect that the model can produce predictions from the combination of the functions that it saw during pretraining. To test this hypothesis in a context with clearly disjoint function classes, we study the ability to perform ICL on linear functions, sinusoids, and convex combinations of the two. We focus on the one dimensional case to make evaluating and visualizing nonlinear function classes straightforward.

Figure 1 shows that while a model pretrained on a mixture of linear functions and sinusoids (i.e. $\mathcal{D}(\mathcal{F}) = 0.5 \cdot \mathcal{D}(\mathcal{F}_{\text{LINEAR}}) + 0.5 \cdot \mathcal{D}(\mathcal{F}_{\text{SINE}})$) is able to make good predictions for either of these functions separately, it is unable to fit a function that is a convex combination of the two[2]. However, it continues to support the narrower hypothesis that when the in-context examples are close to a function class learned in pretraining, the model is capable of selecting the best function class to use for predictions.

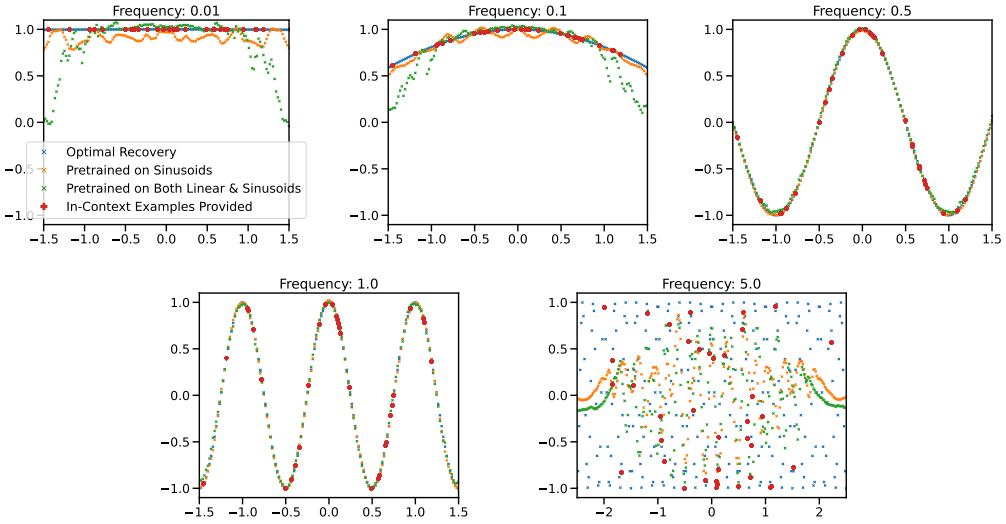
Figure 1c showed a specific convex combination of a linear function and a sinusoid. In Figure 2, we sweep over the relative weights of the linear function and sine wave in the convex combination. Here, we observe that when the combined function is predominantly from one function class or the other – i.e. well-approximated by the function classes learned during pretraining – the in-context predictions are reasonable. However, when both functions contribute significantly to the convex combination, the model makes erratic predictions not well-justified by the in-context examples. This shows that the task-family selection capabilities of the model are limited by proximity to the pretraining data, and suggests that broad coverage of function space is critical for generalized in-context learning capabilities.

The previous convex combinations were specifically constructed so that the model had never seen similar functions in pretraining. Shifting to the scenario where sections of the function class space are increasingly rare in the pretraining data, we find that the model generalization starts strong and then degrades dramatically as the tasks become so rare as to be out-of-distribution. Specifically, we trained models on a mixture of sinusoids with frequencies drawn from a $\text{GAMMA}(6, 1/6)$ distribution. This distribution ensures that the average frequency is 1. Frequencies below 0.01 or above 5 are extremely rare: the probability of a sampling a frequency outside this range is less than 1×10^{-7}, i.e.

[2]Note here convex combination refers to a weighted combination of the function outputs, *not* the weighted mixture distribution used in pretraining which samples f from either either one or the other for the prompt sequence.

we expect to see approximately 100 examples out of the 1 billion used in pretraining. Figure 3 shows the predictions from all of the models on a variety of frequencies inside and outside this range.

Figure 3. Predictions from transformer models pretrained on sinusoids (orange), or both sinusoids and linear functions (green) when provided the red examples in-context, coming from sinusoids of increasing frequency (noted in the plot title).



3 Conclusion

We have empirically explored the role of the pretraining data composition on the ability of pretrained transformers to in-context learn function classes both inside and outside the support of their pretraining data distribution. An important question is understanding how the observations we make here carry over to tokenized models and to questions represented in natural language. We attempted an experiment to train a tokenized model for the one-dimensional examples presented in Section 2 by binning the scalar values into buckets, and treating the bucket indices as tokens input into a transformer-based language model. We trained this model for 5M epochs with a cross-entropy loss as typically used in language models, but were unable to significantly decrease the loss. Understanding the challenges to training such a model and evaluating whether this framing has different task-family selection or out-of-distribution generalization properties is important future work. In the natural language setting, the intuitive notions of how to appropriately define task families (in our case function classes), precise pretraining mixtures, and convex combinations are all less clear. We believe bridging the gap between the notions presented here and in language modeling may help to improve our understanding of the power of ICL and how to effectively enable it.

References

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*, 2022.

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`.

S. Garg, D. Tsipras, P. S. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Y. Li, M. Emrullah Ildiz, D. Papailiopoulos, and S. Oymak. Transformers as algorithms: Generalization and stability in in-context learning. *arXiv e-prints*, pages arXiv–2301, 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`.

J. Von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

R. Zhang, S. Frei, and P. L. Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

A Architecture Training and Data Generation

A.1 Architecture and Training

Following Garg et al. [2022]'s approach to convert a sequence of n alternating $\mathbf{x}$ values and $f(\mathbf{x})$ values into a single embedded sequence for the transformer, we produce a sequence of $2n$ "tokens" of $\mathbf{x}_i$ and $f(\mathbf{x}_i)$, each of which is a real-valued scalar. We insert a (learnable) linear layer to project each input into the transformer's model dimension. Correspondingly, we insert another (learnable) linear layer to reverse the embedding and produce a 1-dimensional output representing $\mathbf{x}$ or $f(\mathbf{x})$, depending on the sequence position. The loss at training time uses the next-token prediction squared loss computed over alternating positions: that is, the loss is only computed over the $\mathbf{x}_i$ positions and ignores the $\mathbf{x}_{i+1}$ value predicted by the transformer structure for each $f(\mathbf{x}_i)$ in the input. As in prior work, the model is pretrained on simulated data for different data-generating setups. We do not fine-tune a pretrained language model and do not train on actual text. 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. $\mathcal{N}(0,1)$) and a single $f \sim \mathcal{D}_{\mathcal{F}}$ per prompt. We use a fixed w per pretraining run.

We trained a decoder-only Transformer [Vaswani et al., 2017] model of GPT-2 scale implemented in the Jax-based machine learning framework, Pax[3] 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 x as 20 (except when specified otherwise) and used standard cosine positional embeddings in our model. We trained 1 million steps with a training batch size of 1024. We used the Adam optimizer with standard values 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. We present our results without noise in the data generation although find similar results adding label noise.

A.2 Data Generation and Function Classes

Throughout our paper we always use $\mathcal{N}(0,1)$ for the covariate distribution (so each $\mathbf{x}_i \sim \mathcal{N}(0,1)$. For the function classes considered in the text we use:

- For the class of linear functions $\mathcal{D}(\mathcal{F}_{\text{LINEAR}})$ we generate a random $\boldsymbol{\beta} \sim \mathcal{N}(0,1)$ and define $\mathcal{D}(\mathcal{F}_{\text{LINEAR}}) = \{f(\mathbf{x}_i) : f(\mathbf{x}_i) = \boldsymbol{\beta}\mathbf{x}_i\}$.
- To sample $f \sim \mathcal{D}(\mathcal{F}_{\text{SINE}})$, first, we sample $\omega \sim \Gamma(6, 1/6)$, then sample $\beta \sim \mathcal{N}(0,1)$, and let $f(x) = \beta \cos(2\pi\omega x)$.

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