CAN TRANSFORMERS LEARN TASKS OF VARYING COMPLEXITY IN-CONTEXT?

Puneesh Deora[1], **Bhavya Vasudeva**[2]*, **Tina Behnia**[1]*, **Christos Thrampoulidis**[1]
[1]University of British Columbia
[2]University of Southern California

ABSTRACT

In-context learning (ICL) is the remarkable ability of trained transformers to adapt to new tasks by leveraging a sequence of examples provided at inference time—without any additional training. Prior work on understanding ICL has primarily focused on setups with fixed task complexity (e.g., linear, logistic, or sinusoidal regression tasks with fixed complexity, and more recently first-order Markov chains), overlooking the diverse range of tasks that large language models encounter in practice. In this paper, we investigate ICL in transformers trained on multiple task categories of varying complexity. Our results show that, during inference, transformers effectively learn in-context by identifying the appropriate task complexity and accurately estimating the corresponding task parameters. We verify our claim with experiments on Markov chains and linear regression tasks of varying complexity. Additionally, our experiments suggest that transformers exhibit a bias towards learning the simplest task that explains the inference-time context.

1 INTRODUCTION

In-context learning (ICL) is a powerful capability of pre-trained large language models (LLMs) that enables them to adapt to new tasks using contextual examples, without requiring any finetuning (Brown et al., 2020). Although much of the interpretability research on ICL has focused on large pre-trained LLMs (Elhage et al., 2021; Wang et al., 2023; Min et al., 2022), controlled, small-scale studies have also been conducted by training transformers from scratch on simpler tasks—such as linear regression (Garg et al., 2022; Akyürek et al., 2023; von Oswald et al., 2022; Zhang et al., 2023), discrete functions (Bhattamishra et al., 2024), and Markov chains (Edelman et al., 2024; Rajaraman et al., 2024; Park et al., 2025). These synthetic setups allow precise control over the training and task distribution, facilitating direct comparisons with known algorithms. However, they are generally limited to tasks of *fixed complexity*, whereas real-world LLMs are pre-trained on large, diverse corpora encompassing tasks of varying complexity.

To bridge this gap, we explore ICL in synthetic environments where tasks exhibit different levels of complexity. In particular, we model various task categories, each representing a fixed complexity level, with the category of highest complexity acting as a superset of all others. Consequently, learning only the most complex category can still yield reasonable performance on lower complexity tasks. In this challenging setup—where categories have varying yet comparable degrees of complexity—we investigate the following key questions:

Can transformers learn tasks of varying complexity learn in-context? Specifically, during inference, can they identify the true task category, or do they simply default to the most complex category?

Figure 1 provides an example using Markov chains: we train a transformer simultaneously on order-1 (the "simpler" category) and order-3 (the "complex" category) chains. We find that at inference time, the transformer learns to recognize the chain order and then predict using either bigram or tetragram statistics, as appropriate. Remarkably, this occurs despite order-1 chains being a special

*Equal contribution.

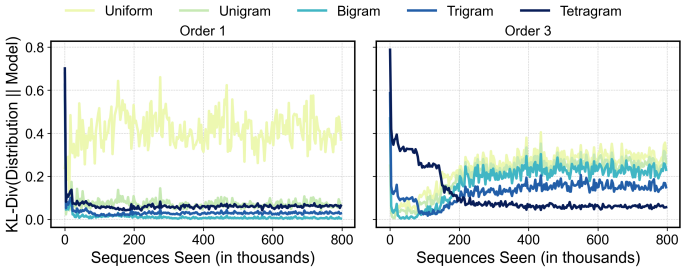

Figure 1: We train a transformer for next-token prediction on sequences generated by random order-1 and order-3 Markov chains. During inference, we assess its ICL ability by evaluating performance on sequences derived from unseen order-1 and order-3 chains. **(Left)** shows the distance between the model's output distribution on the last token and n-gram statistics of the context for order-1 inference, as a function of the number of sequences seen during training. **(Right)** presents analogous results for order-3 inference. *Notably, the trained transformer can identify the true order (1 or 3) of the context, and then predict using either bigram or tetragram statistics accordingly.*

case of order-3 chains, indicating that the model adapts to the context instead of defaulting to using only the higher order Markov chain.

The remainder of the paper is organized as follows. Section 2 discusses related work on ICL in synthetic setups. Section 3 describes how we construct tasks of varying complexity in both Markov chain and linear regression settings. Section 4 presents our experimental results, and Section 5 concludes with a broader discussion. Our contributions are:

- We propose a synthetic framework to study in-context learning for transformers with tasks of varying complexity, extending existing fixed-complexity setups in Markov chains and linear regression to accommodate multiple levels of complexity.
- We show that a transformer trained on these diverse tasks can adaptively identify the correct task category and the underlying parameters during inference.
- We demonstrate that this adaptive capability does not emerge when training exclusively on the most complex tasks, underscoring the importance of task diversity.

2 Previous Work

In this section, we first discuss the key related work on the common synthetic setups in the ICL literature, namely Markov chains and linear regression. Using this, we will set up our corresponding ICL setup with tasks of varying complexity.

2.1 ICL of Markov Chains

Formally, an order-k Markov chain generates an element x_t that is dependent on the previous k entries of the sequence. Mathematically, this is stated as $p(x_t = v|x_{t-1}, ..., x_1) = p(x_t = v|x_{t-1}, ..., x_{t-k+1})$ for all v. All entries v are derived from a vocabulary $\mathcal{V} = \{0, 1, ..., V-1\}$ of size V. For brevity, we refer to this set as $[V]$. These conditional distributions form the rows of the row-stochastic transition matrix $P \in \mathbb{R}^{V^k \times V}$, where each row of P follows a prior β. In previous ICL works (Edelman et al., 2024; Park et al., 2025), it is chosen as the Dirichlet prior, with the parameter $\boldsymbol{\alpha} = (1, ..., 1)^\top$. This corresponds to the case of all sets of transition probabilities being equally probable. The initial k entries of a sequence $\{x_1, ..., x_k\}$ are drawn uniformly at random from the set $[V]$.

Edelman et al. (2024) show that transformers trained on sequences generated from random order-1 Markov chains learn to do in-context inference on unseen Markov chains of order 1. Specifically, let $X_{\leq t} = [x_1, x_2, ..., x_t]$ denote the sub-sequence of X of length t. The transformer M_θ parameterized

by θ is trained to auto-regressively predict x_{t+1} using $X_{\leq t}$ by minimizing the expected loss

$$L(\theta) := \mathop{\mathbb{E}}_{\substack{X\sim P \\ P\sim \mathrm{Dir}(\mathbf{1})^{\otimes V}}} \sum_{t=1}^{T-1} \ell(M_\theta(X_{\leq t}), x_{t+1}), \tag{1}$$

where ℓ is the cross-entropy loss, T denotes the length of each sequence, and $\mathrm{Dir}(\mathbf{1})^{\otimes V}$ denotes V independent draws from the Dirichlet prior with the $\boldsymbol{\alpha}$ set to the all-ones vector. Let $P_{\text{inf}} \sim \mathrm{Dir}(\mathbf{1})^{\otimes V}$ denote the inference time transition matrix, and let $X \sim P_{\text{inf}}$ denote an in-context sequence of length T generated from it. The transformer M_θ trained via the objective in Eq. (1), when given input X, predicts the next token x_{T+1} with a distribution that is close to the bigram probabilities, as measured by the KL divergence. The bigram distribution is as follows:

$$p(x_{T+1} = v|X) = \sum_{t=2}^{T-1} \frac{\mathbb{1}(x_{t-1} = x_T, x_t = v)}{\sum_{t=2}^{T} \mathbb{1}(x_{t-1} = x_T)}, \quad v \in [V]. \tag{2}$$

Note that the transformer learns to do this for all positions x_t, $t \in [T]$, *i.e.*, look back at the sequence, $X_{\leq t}$, compute the bigram probabilities, and predict the next token $v \in [V]$ using these probabilities. In Section 3, we will look at how we extend this setup to Markov chains of varying complexity.

2.2 ICL OF LINEAR REGRESSION

When learning linear regression in-context, the sequences have interleaved $(\mathbf{x}, y)$ pairs of the form $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_t, y_t]$, with vectors $\mathbf{x}_t \in \mathbb{R}^d$, and labels $y_t \in \mathbb{R}$. Vectors $\mathbf{x}_t$ are typically sampled i.i.d Gaussian, *i.e.*, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$. Every sequence is parameterized by a $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, such that labels are given as $y_t = \mathbf{w}^\top \mathbf{x}_t$, $\forall t \in [T]$. Similar to the case of Markov chains, the transformer M_θ is trained to predict label y_{t+1} using the previous t $(\mathbf{x}, y)$ pairs and $\mathbf{x}_{t+1}$, denoted by $X_{\leq t}$, by minimizing the expected loss over sequences with T $(\mathbf{x}, y)$ pairs

$$L(\theta) := \mathop{\mathbb{E}}_{\substack{\mathbf{x}_t\sim\mathcal{N}(\mathbf{0},\mathbb{I}_d) \\ \mathbf{w}\sim\mathcal{N}(\mathbf{0},\mathbb{I}_d)}} \sum_{t=1}^{T-1} \ell(M_\theta(X_{\leq t}), y_{t+1}), \tag{3}$$

where ℓ is the squared loss. The work by Garg et al. (2022); Akyürek et al. (2023) shows that transformers learn linear regression in-context, when they're trained via the above objective. Specifically, given inference-time sequence $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_T, y_T, \mathbf{x}_{\text{test}}]$, with labels y_t generated with an unseen $\mathbf{w}_{\text{inf}} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, the transformer can infer this $\mathbf{w}_{\text{inf}}$ based on the in-context examples in the following sense. They compare the trained transformer's prediction $M_\theta(X)$ to the prediction given by well-known algorithms like least-squares (LS), ridge regression, etc. Using T data-label pairs $\{(\mathbf{x}_t, y_t)\}_{t=1}^T$, they compute the solutions of these algorithms, evaluate them on $\mathbf{x}_{\text{test}}$, and show that the transformer's predictions closely match the LS solution.

3 MODELLING TASKS OF VARYING COMPLEXITY

In the previous section, we examined the ICL capabilities of transformers when trained on tasks of fixed complexity. But what happens when transformers are trained on tasks of varying complexity? Can they estimate the true complexity and the corresponding task parameters at inference time? In this section, we introduce two synthetic settings to model such tasks and explore these questions.

Markov Chains. First, we look at the setup of Markov chains. In order to consider tasks of varying complexity, we look at training the transformer with Markov chains of multiple different orders. Specifically, we create task categories of different complexity by grouping all Markov chains of a fixed order under one task category. Thus, order-1 chains make up one category, order-2 another, and so on. Clearly, the category of order-k_1 Markov chains is of smaller complexity compared to any order-k_2 category ($k_2 > k_1$), as the transition matrices for the latter have higher degrees of freedom, V^{k_2} vs. V^{k_1}, and they can capture higher order relations between the elements of a sequence.

We train the transformer model M_θ on sequences generated from two different task categories[†]. We consider order-1 and order-k categories, where $k > 1$. During training, we first sample a transition

[†]Our results hold for multiple task categories, but for simplicity, we discuss two categories from here on.

matrix P_s of order-s, which is then used to generate a sequence of length $X = [x_1, x_2, ..., x_T]$ of length T. The order s is sampled uniformly at random from the set ord $= \{1, k\}$.

Similar to Eq. (1), the transformer M_θ is trained to auto-regressively predict the element x_{t+1} using $X_{\leq t}$ by minimizing the expected loss, this time over Markov chains of both orders $\{1, k\}$

$$L(\theta) := \mathop{\mathbb{E}}_{\substack{X \sim P_s \\ P_s \sim \text{Dir}(\mathbf{1})^{\otimes V^s} \\ s \sim \text{Unif(ord)}}} \sum_{t=1}^{T} \ell(M_\theta(X_{\leq t}), x_{t+1}), \tag{4}$$

where ℓ is the cross-entropy loss function.

Compared to the fixed order setup in Section 2.1, the above is more challenging. The transformer at inference time, when given an input sequence X from an unseen order-1 or order-k Markov chain, has to identify the true order, and predict based on the inferred order-1 or order-k statistics. Similar to bigram statistics in Eq. (2), order-k statistics are written as

$$p(x_{T+1} = v|X) = \sum_{t=k}^{T-1} \frac{\mathbb{1}(x_{t-1} = x_T, x_{t-2} = x_{T-1}, ..., x_{t-k+1} = x_{T-k}, x_t = v)}{\sum_{t=k}^{T} \mathbb{1}(x_{t-1} = x_T, x_{t-2} = x_{T-1}, ..., x_{t-k+1} = x_{T-k})}. \tag{5}$$

During inference, when prompted with a sequence from an order-k chain, order-k statistics clearly capture the underlying dependencies better compared to order-1 statistics, which are range-limited. On the other hand, all order-1 transition matrices can be written as order-k transition matrices.

*Q: What happens during inference with a sequence generated from order-*1 *transition matrix? Does the transformer predict next-tokens according to order-*1 *or order-*k *statistics?*

In the limit as the context length $T \to \infty$, both statistics converge to the true probabilities of the underlying ground truth transition matrix, making them indistinguishable. But what happens in the regime of finite T, where order-1 statistics diverge from order-k statistics? While higher-order statistics yield a better fit to the observed data (i.e. higher likelihood), they may not always be the most predictive. This raises a key question: during training, does the transformer exclusively learn the highest-order (most complex) task category, or does it develop the ability to distinguish between different underlying orders and adapt accordingly?

Linear Regression. We also construct a similar setup of two task categories for studying ICL of linear regression with varying complexity. As mentioned in Section 2.2, the sequences are of the form $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_T, y_t]$, with vectors $\mathbf{x}_t \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, and labels $y_t \in \mathbb{R}$. For the first category of tasks, $\mathbf{w} = \mathbf{w}_d \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$. For the second category, $\mathbf{w}$ lies on a subspace of $\mathbb{R}^d$ with $\mathbf{w} = [\mathbf{w}_{d/2}, \mathbf{0}]$, where $\mathbf{w}_{d/2} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_{d/2})$. Clearly, the tasks in category one, using $\mathbf{w}_d$, are more complex as they have more degrees of freedom.

Similar to the case of Markov chains, training sequences with T in-context examples are generated as follows: we first sample s uniformly from the set dim $= \{d/2, d\}$, then generate $\mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_s)$, sample vectors $\mathbf{x}_t$, and generate the labels $y_t = \mathbf{w}_s^\top \mathbf{x}_{t,1:s}$, where $t \in [T]$. Here, $\mathbf{x}_{t,1:s}$ denotes the t-th vector with the first s coordinates.

The transformer M_θ is trained to auto-regressively predict the label y_{t+1} using the first t pairs of examples $X_{\leq t}$ by minimizing the expected loss

$$L(\theta) := \mathop{\mathbb{E}}_{\substack{\mathbf{x}_{1:T} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d) \\ \mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_s) \\ s \sim \text{Unif(dim)}}} \sum_{t=1}^{T} \ell(M_\theta(X_{\leq t}; \mathbf{x}_{\text{test}}), y_{t+1}), \tag{6}$$

where ℓ is the squared loss. In this setup with regressors of varying complexity, we can generate an inference-time sequence using $\mathbf{w}_d$ or $\mathbf{w}_{d/2}$. Let X denote the sequence with T in-context examples, and $\mathbf{x}_{\text{test}}$ denote the query for which we want to predict the label.

We consider different benchmark solutions here. We can compare the transformer's performance to the d-dimensional LS solution $\mathbf{w}_d^{\text{LS}}$ or the $d/2$-dimensional LS solution $\mathbf{w}_{d/2}^{\text{LS}} =$

$(X_{d/2}^\top X_{d/2})^{-1} X_{d/2}^\top \mathbf{y}$, where $X_d = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]^\top \in \mathbb{R}^{T \times d}$ the matrix of vectors, and $\mathbf{y} = [y_1, ..., y_T]$ is the vector of labels. Here, $X_{d/2} = X_d[\mathbb{I}_{d/2} \quad \mathbf{0}_{d/2}]^\top$. The predicted label y would be $\mathbf{x}_{\text{test}}^\top \mathbf{w}_d^{\text{LS}}$ in the first scenario and $\mathbf{x}_{\text{test},1:d/2}^\top \mathbf{w}_{d/2}^{\text{LS}}$ in the second one. Note that when the context length $T < d'$ (the dimension of the vectors), LS refers to the minimum ℓ_2-norm interpolating solution.

For a transformer M_θ trained using Eq. (3), we want to examine it with respect to the aforementioned solutions. When prompted with the sequence where $\mathbf{w}_d$ is the underlying regressor, $\mathbf{w}_d^{\text{LS}}$ clearly explains the in-context examples better as $\mathbf{w}_{d/2}^{\text{LS}}$ is limited on a $d/2$-dimensional subspace.

However, *what happens when we prompt the model with a sequence generated with* $\mathbf{w}_{d/2}$*?* For context length $T > d$, $\mathbf{w}_d^{\text{LS}} = \mathbf{w}_{d/2}^{\text{LS}} = \mathbf{w}_{d/2}$. However, this is not necessarily true for $d > T \geq d/2$, where the former is the minimum norm interpolator, and the latter is the true regressor $\mathbf{w}_{d/2}$.

4 RESULTS

We train a GPT-2 type decoder-only transformer for all the experiments (Karpathy, 2023). Please refer to Appendix B for specific training details. We make the following claim, and validate it via experiments with Markov chains and linear regression of varying complexity.

Claim: *During in-context inference, transformers 'learn' the simplest task out of the task categories seen during training.*

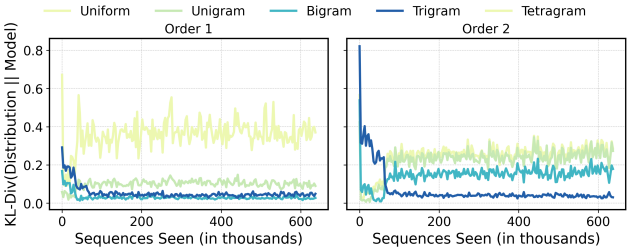

Figure 2: A transformer trained on sequences generated by random order-1 and order-2 Markov chains can infer whether the context is order-1 or order-2, then generate predictions using the corresponding bigram or trigram statistics. **(Left)** shows the distance between the model's output distribution on the last token and well-defined context strategies for order-1 inference, as a function of the number of sequences seen during training. **(Right)** presents analogous results for order-2 inference.

.

Markov Chains. In Fig. 1, we train a transformer on sequences drawn from order-1 and order-3 Markov chains (i.e., Eq. (4) with ord $= \{1, 3\}$). We then measure the KL divergence between the transformer's output distribution and several well-defined strategies when the input is taken from unseen order-1 or order-3 Markov chains. With sufficient training, the transformer learns to distinguish between order-1 and order-3 sequences, using bigram and tetragram statistics, respectively.

In Fig. 2, we repeat this experiment with order-1 and order-2 Markov chains, and again observe that the model learns to infer the underlying order. Notably, in both cases, the transformer does not simply rely on the highest-order statistics (tetragram or trigram), which would have comparable predictive power for order-1 sequences. Instead, it selects the appropriate strategy based on the in-context sequence. Moreover, when prompted with sequences from a higher-order chain (order 3 or 2), the transformer accurately infers that order. These behaviors suggest that the transformer effectively learns the simplest task that explains the observed context.

Linear Regression. Fig. 3 compares the trained transformer's performance with the four benchmarks discussed in Section 3, when trained on sequences drawn from random d-dimensional and $d/2$-dimensional regressors (following the training objective in Eq. (6)). For sequences generated by $\mathbf{w}_d$, the transformer's predictions align more closely with $\mathbf{w}_d^{\text{LS}}$ than with $\mathbf{w}_{d/2}^{\text{LS}}$, as the latter does

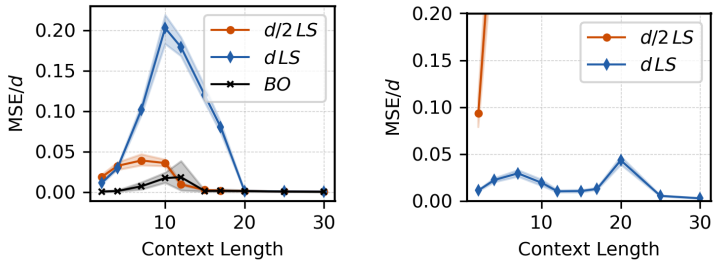

Figure 3: A transformer trained on $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_T, y_t]$ sequences generated using random d-dimensional and $d/2$-dimensional regressors (see Section 3 for details). We plot $(M_\theta([X, \mathbf{x}_{\text{test}}]) - \mathbf{w}_{\text{bench}}^\top \mathbf{x}_{\text{test}})^2/d$ where $\mathbf{w}_{\text{bench}}$ refers to two benchmark least-squares solutions in d or $d/2$ dimensional space described in Section 3. When prompted with sequences from unseen $d/2$ **(left)** or d dimensional regressors **(right)** , the transformer's predictions align most closely with the corresponding LS solution $\mathbf{w}_{d/2}^{\text{LS}}$ or $\mathbf{w}_d^{\text{LS}}$. Here, $d = 20$ and training context length $T = 39$. See App. for a similar experiment with $d = 10$.

not have enough degrees of freedom to fit the data. In contrast, when prompted with sequences from $\mathbf{w}_{d/2}$ in the $d > T \geq d/2$ regime, the transformer's predictions are closer to $\mathbf{w}_{d/2}^{\text{LS}}$ (the true regressor) than to $\mathbf{w}_d^{\text{LS}}$. Unlike the $\mathbf{w}_d$ case—where only $\mathbf{w}_d^{\text{LS}}$ can adequately fit the data—both $\mathbf{w}_d^{\text{LS}}$ and $\mathbf{w}_{d/2}^{\text{LS}}$ fit the context here, yet the transformer favors the simpler predictor $\mathbf{w}_{d/2}^{\text{LS}}$. Finally, once $T \geq d$, we have $\mathbf{w}_d^{\text{LS}} = \mathbf{w}_{d/2}^{\text{LS}} = \mathbf{w}_{d/2}$, and the transformer nearly recovers the true regressor.

Training with only the complex task. We also train the transformer on fixed order-k Markov chains (following Edelman et al. (2024); Park et al. (2025)) to examine whether it can infer the order when presented with sequences generated from a lower-order chain ($< k$) at inference time. Fig. 4 explores this and shows that this is not the case. This finding is crucial, as it suggests that a transformer trained on an order-k chain learns only the order-k statistics of the context and does not generalize to lower-order statistics. We also do this experiment for the case of linear regression (see Fig. 6 in the App.) and observe a similar behaviour.

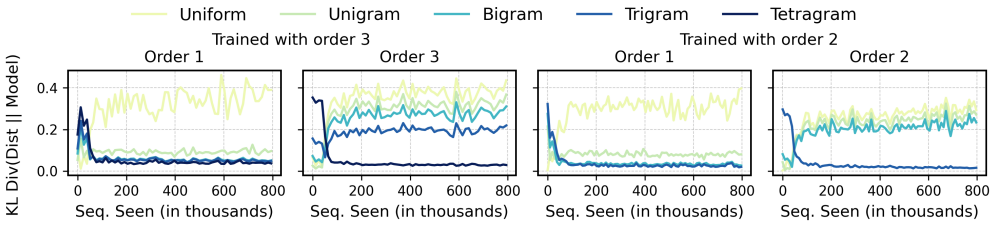

Figure 4: A transformer trained on random order-3 (left column) or order-2 (right column) Markov chains can only predict based on order-3 (left column) or order-2 (right column) statistics, and fails to predict based on order-1 statistics when given order-1 in-context sequences.

5 Discussion

In this work, we examine the in-context learning capabilities of transformers when trained on tasks of varying complexity. We model this by defining multiple task categories, each representing a different level of complexity, with the highest category acting as a superset of the others. Our experiments show that under these conditions, transformers learn to identify the true task complexity and associated parameters during inference, rather than simply defaulting to the highest-complexity category. This behavior suggests an inherent bias toward selecting the simplest task that adequately explains the observed context.

As the transformer can effectively "switch" between task categories at inference, we hypothesize that an implicit Bayesian inference perspective can explain this. Prior work on ICL as implicit Bayesian inference (Xie et al., 2022; Panwar et al., 2024) posits that during pre-training, LLMs learn a prior over tasks, and at inference, they apply this prior based, and update their posterior based on in-context examples to "learn" a new task. In our setup, the bias toward the simplest category can be interpreted as assigning a higher prior to simpler tasks. For instance, in the Markov chains setting of training with order-1 and order-3 chains, this means that $p(\text{ord}_1) > p(\text{ord}_3)$. During order-1 inference, although the likelihood for tetragram, $p(X \mid \text{ord}_3) > p(X \mid \text{ord}_1)$ (the likelihood for bigram), the strong prior favoring ord_1 leads to a higher posterior for order-1, aligning the output with bigram statistics. In contrast, for order-3 inference, the likelihood $p(X \mid \text{ord}_3)$ is sufficiently larger than $p(X \mid \text{ord}_1)$ to overcome the prior gap, prompting the model to use tetragram statistics instead. Future work includes finding more concrete evidence for this hypothesis, such as constructing a precise example of how the transformer 'switches' between task categories.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *Int. Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ekeyCgeRfC.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Ezra Edelman, Nikolaos Tsilivis, Benjamin L. Edelman, Eran Malach, and Surbhi Goel. The evolution of statistical induction heads: In-context learning markov chains. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qaRT6QTIqJ.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=flNZJ2eOet.

A. Karpathy. Mingpt. https://github.com/karpathy/minGPT/tree/master, 2023.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

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

Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8298–8319, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.527. URL https://aclanthology.org/2023.findings-acl.527/.

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

Core Francisco Park, Ekdeep Singh Lubana, and Hidenori Tanaka. Algorithmic phases of in-context learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XgH1wfHSX8.

Nived Rajaraman, Marco Bondaschi, Ashok Vardhan Makkuva, Kannan Ramchandran, and Michael Gastpar. Transformers on markov data: Constant depth suffices. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=5uG9tp3v2q.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: An information flow perspective for understanding in-context learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9840–9855, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.609. URL https://aclanthology.org/2023.emnlp-main.609/.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu 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.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context, 2023.

A Additional Results

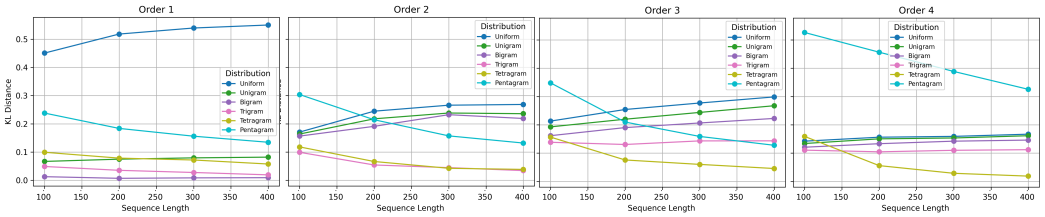

Figure 5: Figure illustrates how varying the context length affects a trained transformer's performance when prompted with sequences from different-order chains at inference. This transformer was trained on sequences from order-1 and order-3 chains (Fig. 1). Here, we also evaluate its performance on order-2 and order-4 sequences, which were not seen during training. Context length plays a crucial role: while the transformer can accurately estimate the true order for order-1 and predict using bigram statistics, a longer context is required to recognize order-3 sequences. This observation aligns with the notion of in-context "learning" versus "retrieval" noted in prior work (Park et al., 2025; Lin & Lee, 2024; Pan et al., 2023), suggesting that for smaller context lengths, the transformer primarily "retrieves" tasks it has seen during training, whereas longer contexts allow it to "learn" tasks that it has not previously encountered. For order-2 sequences, the transformer's predictions lie between trigram and tetragram strategies, whereas for order-4 sequences, they most closely match tetragram statistics. This outcome is expected because the model was not trained on tasks of ; it applies the strategy that best explains the observed context at inference time.

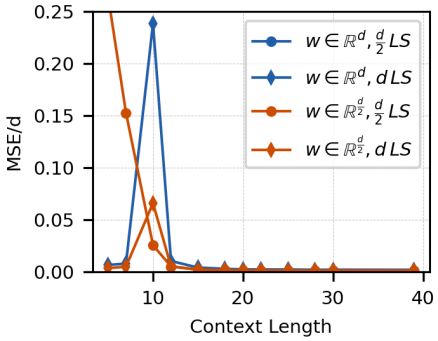

Figure 6: A transformer trained on $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_T, y_t]$ sequences generated using fixed complexity, random d-dimensional regressors $\mathbf{w}_d$ (see Section 3 for details). We plot $(M_\theta([X, \mathbf{x}_{\text{test}}]) - \mathbf{w}_{\text{bench}}^\top \mathbf{x}_{\text{test}})^2/d$ where $\mathbf{w}_{\text{bench}}$ refers to two benchmark least-squares solutions in d or $d/2$ dimensional space described in Section 3. We see that the transformer's predictions align most closely with $\mathbf{w}_d^{\text{LS}}$ no matter the type of inference-time regressor used to generate the sequences ($\mathbf{w}_d$ or $\mathbf{w}_{d/2}$). This indicates that the transformer doesn't learn to estimate the lower-complexity solution $\mathbf{w}_{d/2}^{\text{LS}}$, unlike the case in Fig. 3. Here, $T = 39$ and $d = 10$. Also the first curve (blue, dot) is out of the plotted range.

B Details of Experimental Settings

For both sets of experiments, we train GPT-2 type decoder-only transformer Karpathy (2023). We use AdamW Loshchilov & Hutter (2019) optimizer in all experiments with a learning rate of $1e-4$.

Markov chains. For all Markov chain experiments, vocab size $V = 3$. For the order-1 and order-3 experiments in Fig. 1, context length (T) for all sequences was set 300. We used a 6 layer, 6 head transformer, with embedding dimension set to 192. For the order-1 and order-2 experiments in

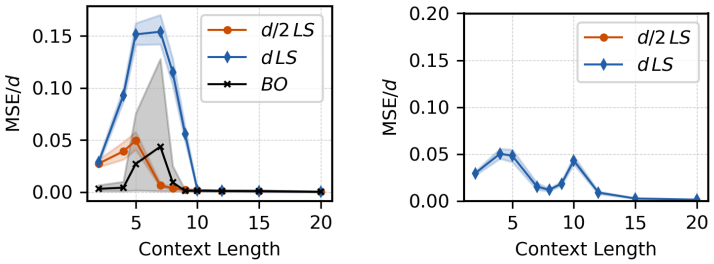

Figure 7: Replicating the experiment in Fig. 3 using $d = 10$.

Fig. 2, context length T was set to 200. We used a 2 layer transformer with 20 heads, and embedding dimension was set to 320. In all sets of experiments, we used a batch size of 32. Additionally, we used relative position encoding in all the experiments. For the fixed-order experiments in Fig. 4, we used the setup from the corresponding variable order experiment.

Linear regression. We used a 12 layer, 8 head transformer with embedding dimension $= 256$ similar to Garg et al. (2022). The batch size was set to 32, with $d = 20$ and $d = 20$ for Fig. 3 and Fig. 7, respectively. The training context length T is set to 39 in both experiments.