In-Context Occam's Razor: How Transformers Prefer Simpler Hypotheses on the Fly

Puneesh Deora [1] **Bhavya Vasudeva** [2] **Tina Behnia** [1] **Christos Thrampoulidis** [1]

Abstract

In-context learning (ICL) enables transformers to adapt to new tasks through contextual examples without parameter updates. While existing research typically studies ICL in fixed-complexity environments, language models encounter tasks spanning diverse complexity levels. We investigate how transformers navigate hierarchical task structures where higher-complexity categories can perfectly represent any pattern generated by simpler ones. Specifically, we design testbeds of Markov-chain and linear-regression data that reveal transformers not only identify the appropriate complexity level for each task but also accurately infer the corresponding parameters—even when the in-context examples are compatible with multiple complexity hypotheses. When presented with data generated by simpler processes, transformers consistently favor the least complex sufficient explanation. We theoretically explain this behavior through a Bayesian framework, demonstrating that transformers effectively implement an in-context Bayesian Occam's razor by balancing model fit against complexity penalties.

1. Introduction

In-context learning (ICL) has emerged as a fundamental capability of large language models (LLMs), enabling them to adapt to novel tasks through contextual examples without parameter updates (Brown et al., 2020). This remarkable ability has drawn significant attention in interpretability research, with studies examining both commercial-scale LLMs (Elhage et al., 2021; Wang et al., 2023; Min et al., 2022) and controlled, synthetic environments. The latter approach trains transformers from scratch on well-defined 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 processes (Edelman et al., 2024; Rajaraman et al., 2024; Park et al., 2025)—offering precise control over training distributions and enabling direct comparisons with known algorithms. However, these synthetic studies typically restrict analysis to tasks of *fixed complexity*, diverging from real-world scenarios where LLMs encounter diverse tasks spanning multiple complexity levels.

To bridge this gap, we investigate ICL in environments featuring hierarchical task-complexity structures. We design task categories with *distinct complexity* levels, where higher-complexity ones form strict supersets of lower ones: specifically, the higher-complexity category contains hypotheses capable of perfectly emulating any pattern produced by simpler categories. For instance, consider a transformer trained on next-token prediction for sequences generated by both 1st-order and 3rd-order Markov chains. Since any order-1 chain can be perfectly represented as a special case of order-3, an inherent ambiguity arises during inference: When presented with a sequence genuinely generated by a 1st-order chain, can the transformer identify the true complexity class, or will it default to the most expressive hypothesis in its repertoire? More generally:

Can transformers effectively differentiate between tasks of varying complexity during in-context learning? When presented with data compatible with multiple hypothesis classes, do they accurately identify the simplest sufficient hypothesis, or do they systematically default to the most complex available hypothesis?

Our systematic investigation answers this question affirmatively. For example, Figure 1 demonstrates that at inference time, the transformer successfully recognizes the true order of the generating chain. When presented with an order-1 chain, it

[1]University of British Columbia, Vancouver, Canada [2]University of Southern California, Los Angeles, USA. Correspondence to: Puneesh Deora <deora.puneesh@gmail.com>.

Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025, Vancouver, Canada.

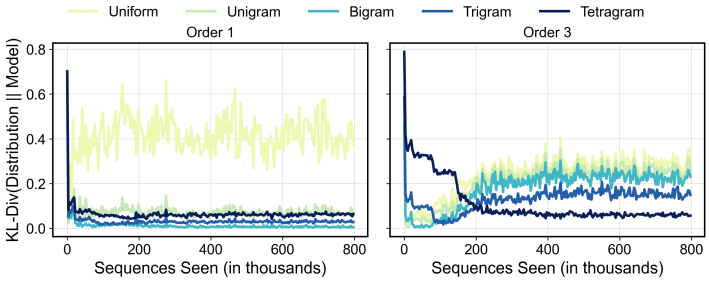

Figure 1. We train a transformer for next-token prediction on sequences generated by random order-1 and order-3 Markov chains. During inference, we assess its ICL ability by evaluating performance on sequences derived from unseen order-1 and order-3 chains. **(Left)** shows the distance between the model's 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.*

appropriately employs bigram statistics; when presented with an order-3 chain, it switches to tetragram statistics.

Our contributions are as follows:

- We introduce a framework for studying ICL across hierarchical complexity levels, where higher-complexity tasks form strict supersets of simpler ones, creating inherent ambiguity in hypothesis selection in context. Concretely, we study controlled synthetic environments with well-defined tasks focusing on Markov chains Edelman et al. (2024) and linear regression (Garg et al., 2022). Unlike prior works that focus on tasks on fixed complexity (see Appendix A), we introduce two extensions—to systematically investigate behaviors when training on tasks spanning multiple complexity levels. In Appendix C.1, we further verify the conclusions on a probabilistic grammar setting.
- We empirically demonstrate in both Markov chain and linear regression settings that transformers correctly identify the simplest sufficient hypothesis rather than defaulting to more expressive models.
- We provide a theoretical justification through a Bayesian lens, showing that transformers naturally implement Bayesian Occam's razor in-context by balancing data likelihood against model complexity.

2. In-Context Learning Across Hierarchical Complexity Levels

2.1. Markov Chains

Modeling Tasks of Varying Complexity. We train on Markov chains of different orders (vocab size V), where we group all chains of a fixed order into one task category. The category of order-k_1 has lower complexity than any order-k_2 category for $k_2 > k_1$ due to fewer degrees of freedom: any order-k_1 chain can be perfectly represented as a special case of an order-k_2 chain, making higher-order categories strict supersets of lower-order ones.

For concreteness, we train the transformer model M_θ on sequences generated from two different task categories[1]. We consider order-1 and order-k categories, where $k > 1$. During training, we first sample a transition matrix P_s of order-s, which is then used to generate a sequence $X = [x_1, x_2, ..., x_T]$ of length T. The order s is sampled uniformly at random from the set ord $= \{1, k\}$. We train autoregressively to minimize the (expected) loss, over Markov chains of both orders $\{1, k\}$, with cross-entropy loss ℓ (see Appendix B.1 for additional details).

At inference, we prompt the model with a sequence generated from either order-1 (simple) or order-k (complex) chain. To evaluate the ICL ability of the model, we compare (with respect to KL divergence) the model's output probability for the token following the prompt to either the order-1 statistics or the order-k statistics. When prompted with an order-k chain, the transformer must recognize the higher-order dependencies that cannot be captured by order-1 statistics. However, when prompted with an order-1 chain, the transformer faces a more subtle decision, as any order-1 transition matrix can be perfectly represented as a special case of an order-k transition matrix. This latter scenario directly tests whether the model defaults to the most complex hypothesis or correctly identifies the simplest sufficient one.

[1]Multiple task categories can be treated similarly without further insights. We focus on two categories for simplicity of exposition.

Transformers Distinguish Between Task Categories In-context. We train a GPT-2 type decoder-only transformer for all the experiments (Karpathy, 2023); Appendix E for details.

In Figure 1, we train a transformer on sequences drawn from order-1 and order-3 Markov chains. We then measure the KL divergence between the transformer's output distribution and several well-defined statistical strategies when the input is taken from unseen order-1 or order-3 Markov chains: uniform, bigram, trigram, tetragram statistics ($k=0,\ldots,3$). We see that with sufficient training, the transformer consistently learns to distinguish between order-1 and order-3 sequences, applying bigram and tetragram statistics appropriately to each case. We validate this finding in Figure 3 in Appendix C by repeating the experiment with order-1 and order-2 Markov chains, again observing that the model accurately infers the underlying order. Crucially, in both experiments, the transformer does not default to the highest-order statistics (tetragram or trigram) for all inputs, despite these statistics having sufficient expressive power to model order-1 sequences. Instead, it selects the appropriate complexity level based on the in-context sequence. This demonstrates that the transformer effectively implements Occam's razor, identifying the simplest hypothesis adequately explaining prompt.

In Appendix C, we provide additional experiments examining how performance varies with context length.

Bayesian Interpretation of the Selection Mechanism. We adapt a Bayesian viewpoint of ICL (Panwar et al., 2024; Xie et al., 2022; Lin & Lee, 2024) to explain how transformers choose the simplest task that explains the context. In this respect, we assume sufficient training data such that the transformer implements the Bayes optimal minimizer. Let $X=[x_1,\ldots,x_T]$ be an observed sequence for $T \gg k$. Then, the *Bayes' optimal* predictive distribution for x_{T+1} given X is the mixture

$$\underbrace{p(x_{T+1}=v \mid X)}_{\text{model output}} = \sum_s \underbrace{p(s \mid X)}_{\text{category posterior}} \cdot \underbrace{p(x_{T+1}=v \mid X, s)}_{s\text{-gram statistics of } X}, \quad v \in [V], \tag{1}$$

where $p(s \mid X)$ denotes the posterior probability of the order-s chain given the sequence. Further, $p(x_{T+1}=v \mid X, s)$ denotes the order-s Bayes' optimal predictive distribution. Specifically, for Dirichlet prior $\mathrm{Dir}(\mathbf{1})^{\otimes V^s}$ it can be shown that $p(x_{T+1}=v|X\,,s)$ which is a smoothed version of the order-s statistics; e.g. see Edelman et al. (2024, Eq. (3)). When $T \gg k$, where both statistics are well-defined for both orders 1 and k, the smoothing effect becomes negligible.

From Equation (1), the model's output (LHS) when trained with chains of all orders s can be seen as a *convex mixture* of what would have been the model's output if the transformer was trained only on a single order. This means the model's output crucially depends on the posterior probability coefficients $p(s \mid X)$, which we can express as: $p(s \mid X) = p(X \mid s)\,\pi(s) \big/ \sum_{s' \in \text{ord}} p(X \mid s')\,\pi(s')$. Here, $\pi(s)$ is the prior on orders, which is set to be uniform and thus cancels out in the ratio, and $p(X \mid s)$ is the marginal likelihood of the prompt (context) given the order s. Consequently, which category's statistics (s-gram) the transformer's outputs depends on depends on which marginal likelihood $p(X \mid s)$ dominates.

Asymptotics of Likelihood. For large $T \gg k$, standard Dirichlet–multinomial conjugacy combined with a Bayesian Information Criteria (BIC) style Laplace approximation (Schwarz, 1978; Ghosal & van der Vaart, 2017; Csiszár & Talata, 2006) gives the following useful approximation to the marginal likelihood (Schwarz, 1978):

$$\log p(X \mid s) \;\approx\; \sum\nolimits_t \log \hat{p}_X(x_t \mid x_{t-1}, ..., x_{t-s}) \;-\; \tfrac{V^s(V-1)}{2} \log T\,. \tag{2}$$

Here, $\hat{p}_X$ denotes the *empirical* conditional probabilities derived from the prompt X. See Appendix B.2 for details. Thus, the marginal likelihood decomposes into an empirical likelihood term and a model complexity penalty term.

Bayesian Occam's Razor. Suppose the data is generated by an order-s^* Markov chain, representing a simple category. We wish to compute the ratio $p(X \mid s)/p(X \mid s^*)$ for any complex category of order-s with $s > s^*$. Note that for $T \gg k$, for any $s \geq s^*$ the empirical probabilities $\hat{p}_X(x_t \mid x_{t-1}, \ldots, x_{t-s})$ approach the true data transition probabilities derived from the simple category s^*. Thus, the empirical likelihood terms in Eqn. (9) are (approximately) equal for any $s \geq s^*$. However, the model complexity penalty term being proportional to V^s strongly favors the smaller s. Together, for $T \gg k$ it holds:

$$\forall s > s^* : \frac{p(X \mid s)}{p(X \mid s^*)} \approx 0 \;\Rightarrow\; p(s^* \mid X) \approx 1 \overset{Eqn.(\,1)}{\Longrightarrow} \underbrace{p(x_{T+1} = \cdot \mid X \sim P_{s^*})}_{\substack{\text{model output conditioned on} \\ \text{prompt from simple category}}} \approx \overset{s^*\text{-gram}}{\text{statistics}}. \tag{3}$$

The model implements *Bayesian Occam's razor* in favor of the true lower-order model s^*.

In App. B.2, we show that the when the true process is order-k ($k > 1$) and T is large, then the posterior probability *saturates* to the true order. In App. D, we construct a 2-layer transformer that can compute the empirical conditional probabilities $\hat{p}_X$.

2.2. Linear Regression

Modeling Tasks of Varying Complexity. We design a similar two-category linear-regression setup with hierarchical complexity. The complex category draws weights $\mathbf{w} = \mathbf{w}_d \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ over all d features, whereas the simple category restricts to a $d/2$-dimensional subspace: $\mathbf{w} = [\mathbf{w}_{d/2}, \mathbf{0}]$ with $\mathbf{w}_{d/2} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_{d/2})$[2]. Thus every simple task is a sparse special case of the complex one. For training, we first sample s uniformly from the set $\text{dim} = \{d/2, d\}$, then generate $\mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_s)$ and generate the labels $y_t = \mathbf{w}^\top \mathbf{x}_t$ for $t \in [T]$ where for $s = d/2$, we effectively use $\mathbf{w} = [\mathbf{w}_s, \mathbf{0}]$ (zero-padded to dimension d). See Appendix B.3 for details.

At inference, we generate prompts using either $\mathbf{w}_d$ or $\mathbf{w}_{d/2}$ as the underlying task parameters. Given in-context examples $X = (\mathbf{x}_t, y_t)_{t=1}^T$ and query $\mathbf{x}_{\text{test}}$, we compare the transformer's prediction to two least-squares (LS) benchmark estimates: $\hat{y} = \mathbf{x}_{\text{test}}^\top \mathbf{w}^{\text{LS}}$, where $\mathbf{w}^{\text{LS}}$ is either the full d-dimensional solution $\mathbf{w}_d^{\text{LS}} := A_d^\dagger \mathbf{y}$ or the restricted $d/2$-dimensional solution $\mathbf{w}_{d/2}^{\text{LS}} = A_{d/2}^\dagger \mathbf{y}$. Here, $A_d := [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]^\top$ is the feature matrix, $\mathbf{y} = [y_1, \ldots, y_T]$ is the label vector, and $A_{d/2} := A_d [\mathbb{I}_{d/2} \quad \mathbf{0}_{d/2}]^\top$ is the projection of the feature matrix onto the first $d/2$ dimensions. $A^\dagger$ denotes the Moore-Penrose pseudoinverse. When context length $T < d$ or $T < d/2$ for the respective estimates, the LS solution corresponds to the minimum ℓ_2-norm interpolating solution.

When prompted with a sequence generated by the complex regressor $\mathbf{w}_d$, $\mathbf{w}_d^{\text{LS}}$ naturally provides a better fit than the dimensionality-constrained $\mathbf{w}_{d/2}^{\text{LS}}$. However, the critical question arises when the data comes from $\mathbf{w}_{d/2}$: Does the transformer default to the more expressive solution ($\mathbf{w}_d^{\text{LS}}$), or does it correctly identify the simpler hypothesis? Specifically, when $d > T \geq d/2$, both solutions perfectly interpolate the context data, yet they generally differ in their predictions. Hence, it cleanly tests whether transformers implement a form Occam's razor in-context.

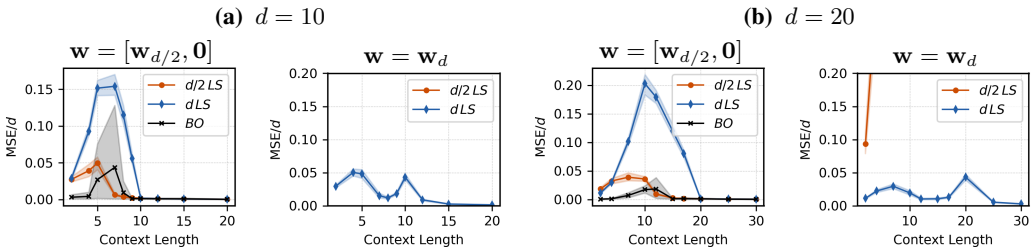

Figure 2. A transformer trained on $T = 39$-long sequences generated using random **(a)** $d = 10$ or **(b)** $d = 20$ dimensional regressors. We plot $(M_\theta([X, \mathbf{x}_{\text{test}}]) - \mathbf{w}^\top \mathbf{x}_{\text{test}})^2/d$ for the two LS baselines $\mathbf{w}_{d/2}^{\text{LS}}, \mathbf{w}_d^{\text{LS}}$ (Section 2). When prompted with unseen sequences of **(left)** $d/2$ or **(right)** d dimensional regressors, the predictions align most closely with the matching baseline. In the right panels the orange dot (off-scale) highlights $\mathbf{w}_{d/2}^{\text{LS}}$'s poor fit to $\mathbf{w}_d$ data. See Appendix C for evaluation over train time.

Transformers Distinguish Between Task Categories In-context. Figure 2 demonstrates how transformers distinguish between task complexities in linear regression. Given data generated from the complex regressor $\mathbf{w}_d$, predictions follow the full LS solution $\mathbf{w}_d^{\text{LS}}$, since $\mathbf{w}_{d/2}^{\text{LS}}$ is under-parametrised. The key test is simple data $\mathbf{w}_{d/2}$ with $d > T \geq d/2$: although $\mathbf{w}_d^{\text{LS}}$ and $\mathbf{w}_{d/2}^{\text{LS}}$ both interpolate the context, the model consistently favours $\mathbf{w}_{d/2}^{\text{LS}}$, a clear demonstration of Occam's razor at work. As context length increases into the $T \geq d$ regime, we observe $\mathbf{w}_d^{\text{LS}} = \mathbf{w}_{d/2}^{\text{LS}} = \mathbf{w}_{d/2}$, and the transformer essentially recovers the true regressor. In Appendix B.4, we explain the preference for the simpler hypothesis using a similar Bayesian viewpoint as done in Section 2.1 for Markov chains.

3. Conclusion

Using simple testbeds, we show that transformers trained on hierarchical task complexity structures implement a form of Bayesian Occam's razor, consistently identifying the simplest sufficient hypothesis without defaulting to more expressive

[2]Choosing dimensionality $d/2$ and aligning the subspace with the coordinate axes are for simplicity, without affecting generality.

model classes. Important questions remain open: From a mechanistic perspective, how do transformers internally implement this complexity selection? From an optimization standpoint, how do the optimization dynamics lead to the emergence of these Bayesian selection principles? Our findings suggest that principled hypothesis selection may be an inherent property of transformers trained on diverse task distributions, potentially contributing to their remarkable generalization capabilities.

References

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=LziniAXEI9.

Ahuja, K., Balachandran, V., Panwar, M., He, T., Smith, N. A., Goyal, N., and Tsvetkov, Y. Learning syntax without planting trees: Understanding hierarchical generalization in transformers, 2025. URL https://arxiv.org/abs/2404.16367.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. 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.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 1, learning hierarchical language structures, 2024. URL https://arxiv.org/abs/2305.13673.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=liMSqUuVg9.

Bhattamishra, S., Patel, A., Blunsom, P., and Kanade, V. 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.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (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.

Cai, T. T., Liang, T., and Zhou, H. H. Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions. *Journal of Multivariate Analysis*, 137:161–172, 2015.

Chen, S., Sheen, H., Wang, T., and Yang, Z. Unveiling induction heads: Provable training dynamics and feature learning in transformers. *arXiv preprint arXiv:2409.10559*, 2024.

Csiszár, I. and Talata, Z. Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE Transactions on Information Theory*, 52(3):1007–1016, 2006. doi: 10.1109/TIT.2005.864420.

Edelman, E., Tsilivis, N., Edelman, B. L., Malach, E., and Goel, S. 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.

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

Elmoznino, E., Marty, T., Kasetty, T., Gagnon, L., Mittal, S., Sridhar, D., and Lajoie, G. In-context learning and occam's razor, 2025. URL https://openreview.net/forum?id=2PKLRmU7ne.

Fu, D., qi Chen, T., Jia, R., and Sharan, V. Transformers learn to achieve second-order convergence rates for in-context linear regression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L8h6cozcbn.

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

Ghosal, S. and van der Vaart, A. *Fundamentals of Nonparametric Bayesian Inference*, volume 44 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, 2017. doi: 10.1017/9781139029834.

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

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: generalization and stability in in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Li, Y., Rawat, A. S., and Oymak, S. Fine-grained analysis of in-context linear estimation: Data, architecture, and beyond. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lYPAYmfQqm.

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

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Exposing attention glitches with flip-flop language modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=VzmpXQAn6E.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=De4FYqjFueZ.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.

Lu, Y. M., Letey, M. I., Zavatone-Veth, J. A., Maiti, A., and Pehlevan, C. Asymptotic theory of in-context learning by linear attention, 2025. URL https://arxiv.org/abs/2405.11751.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? In Goldberg, Y., Kozareva, Z., and Zhang, Y. (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/.

Pan, J., Gao, T., Chen, H., and Chen, D. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (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/.

Panwar, M., Ahuja, K., and Goyal, N. In-context learning through the bayesian prism, 2024. URL https://arxiv.org/abs/2306.04891.

Park, C. F., Lubana, E. S., and Tanaka, H. Algorithmic phases of in-context learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XgH1wfHSX8.

Rajaraman, N., Bondaschi, M., Makkuva, A. V., Ramchandran, K., and Gastpar, M. 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.

Raventos, A., Paul, M., Chen, F., and Ganguli, S. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BtAz4a5xDg.

Schwarz, G. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

Tulino, A. M., Verdú, S., et al. Random matrix theory and wireless communications. *Foundations and Trends® in Communications and Information Theory*, 1(1):1–182, 2004.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Wang, L., Li, L., Dai, D., Chen, D., Zhou, H., Meng, F., Zhou, J., and Sun, X. Label words are anchors: An information flow perspective for understanding in-context learning. In Bouamor, H., Pino, J., and Bali, K. (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/.

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

Zhang, R., Frei, S., and Bartlett, P. L. Trained transformers learn linear models in-context, 2023.

Zhang, R., Wu, J., and Bartlett, P. In-context learning of a linear transformer block: benefits of the mlp component and one-step gd initialization. *Advances in Neural Information Processing Systems*, 37:18310–18361, 2024.

A. Background and Related Works

A.1. Markov Chains and Linear Regression as Testbeds for ICL

ICL of Markov Chains. Markov chains were introduced as a testbed for studying ICL by Edelman et al. (2024). Specifically, they showed 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 by learning to output bigram statistics of the context.

To establish the foundation for our study, we now formalize their key findings. Let x_t denote the t-th symbol from a finite vocabulary $[V] = \{1, ..., V\}$ of size V generated by a k-th order Markov chain, i.e., $p(x_t = v|x_{t-1}, ..., x_1) = p(x_t = v|x_{t-1}, ..., x_{t-k+1})$ for all $v \in [V]$ and let $P \in \mathbb{R}^{V^k \times V}$ denote the row-stochastic transition matrix with these conditional distributions as rows. Sample a transition matrix at random: for example, sample each of its rows according to a Dirichlet prior with parameter $\boldsymbol{\alpha}$. Generate a sequence $X := X_{\leq T} = [x_1, x_2, ..., x_{T-1}]$ of $T \gg k$ symbols from this transition matrix by letting the first k entries $\{x_1, ..., x_k\}$ be drawn uniformly at random. A transformer model M_θ parameterized by θ is trained to auto-regressively predict the next-symbol x_{t+1} using the previous t symbols $X_{\leq t}$ by (approximately) minimizing the expected loss, with respect to cross-entropy ℓ:

$$L(\theta) := \mathbb{E}_{X \sim P \sim \text{Dir}(\mathbf{1})^{\otimes V}} \left[\sum_{t=1}^{T-1} \ell(M_\theta(X_{\leq t}), x_{t+1}) \right], \tag{4}$$

where $\text{Dir}(\mathbf{1})^{\otimes V}$ denotes V independent draws from the Dirichlet prior. At inference, draw transition matrix $P_{\text{inf}} \sim \text{Dir}(\mathbf{1})^{\otimes V}$, and let $X \sim P_{\text{inf}}$ denote a sequence of length $T-1$ generated from it that is given as prompt to the model. Edelman et al. (2024) show that the Kullback–Leibler (KL) divergence of the model's output probability to the birgram statistics

$$\sum_{t=2}^{T-1} \mathbb{1}(x_{t-1} = x_T, x_t = v) \Big/ \sum_{t=2}^{T} \mathbb{1}(x_{t-1} = x_T), \quad v \in [V], \tag{5}$$

averaged over many realizations of X, P_{inf}, is (almost) zero. Here, $\mathbb{1}(\cdot)$ is the indicator function that equals 1 when the condition is true and 0 otherwise. Thus, the transformer looks back at the sequence to compute the bigram probabilities, and predict the next token $v \in [V]$ using these probabilities.

ICL of Linear Regression. Before Markov chains, the first synthetic playground for ICL was introduced by Garg et al. (2022), who demonstrated that a transformer trained on random examples from linear regression tasks can learn to perform in-context inference on unseen tasks by computing the least-squares solution. In this setup, sequences consist of interleaved input-output pairs $(\mathbf{x}, y)$ of the form $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_t, y_t]$, with feature vectors $\mathbf{x}_t \in \mathbb{R}^d$ and labels $y_t \in \mathbb{R}$. The feature vectors $\mathbf{x}_t$ are sampled i.i.d from a standard Gaussian distribution, *i.e.*, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$. Each sequence is characterized by a task-specific weight vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, such that labels are generated as $y_t = \mathbf{w}^\top \mathbf{x}_t, \forall t \in [T]$.

Similar to the Markov chain setup, the transformer M_θ is trained to predict label y_{t+1} using the previous t pairs $(\mathbf{x}, y)$ along with the new input $\mathbf{x}_{t+1}$ (collectively denoted by $X_{\leq t}$), by minimizing the expected loss over sequences with T pairs:

$$L(\theta) := \mathbb{E}_{\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d),\, \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)} \left[\sum_{t=1}^{T-1} \ell(M_\theta(X_{\leq t}), y_{t+1}) \right], \tag{6}$$

where ℓ is the squared loss. Garg et al. (2022) demonstrated that when presented with an inference-time sequence $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_{T-1}, y_{T-1}, \mathbf{x}_T]$ generated from an unseen task vector $\mathbf{w}_{\text{inf}} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, the transformer effectively infers $\mathbf{w}_{\text{inf}}$ from the in-context examples. Specifically, the squared error between the model's prediction and the least-squares prediction $\hat{y}_{\text{LS}} = \mathbf{x}_T^\top \mathbf{w}_{\text{LS}}$ where $\mathbf{w}_{\text{LS}} = A^\dagger \mathbf{y}$ approaches zero when averaged over many realizations of X and $\mathbf{w}_{\text{inf}}$. Here, $A \in \mathbb{R}^{(T-1) \times d}$ represents the feature matrix with columns $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T-1}]^\top$, $\mathbf{y} \in \mathbb{R}^{T-1}$ is the vector of context labels $[y_1, \ldots, y_{T-1}]$, and $A^\dagger$ denotes the Moore-Penrose pseudoinverse.

A.2. Related Work

Task Diversity: Learning vs Retrieval. In the setups investigated by Garg et al. (2022); Edelman et al. (2024), the transformer is trained on sequences generated from fresh tasks drawn from the appropriate distributions at each optimization iteration. Raventos et al. (2023); Park et al. (2025) have further extended this study for linear regression and Markov chains respectively, examining scenarios where the model is trained on sequences generated from a finite number of tasks, thus modeling and analyzing the impact of task diversity in ICL and revealing a tradeoff between two distinct modes of ICL: task retrieval and task learning (Pan et al., 2023). See also Lu et al. (2025) for a high-dimensional statistical approach that theoretically justifies some of these empirical findings. In this work, we focus on the learning mechanism of ICL and

maintain the vanilla setting of training with fresh tasks at each iteration. However, in our settings these tasks will originate from a finite number of complexity categories rather than a single one.

Theoretical Explanations: Linear Regression and Markov Chains. One of the first works investigating the theoretical foundations of ICL is Xie et al. (2022), which proposed a Bayesian perspective that has informed numerous subsequent studies, e.g., (Raventos et al., 2023). Most relevant to our work, Edelman et al. (2024) demonstrated that the bigram statistics rule implemented by transformers trained on order-1 Markov chains is the Bayes' optimal solution (given the context) under the Dirichlet prior. Similarly, for linear regression, the least-squares solution learned by transformers is the Bayes' optimal predictor (given the context) under Gaussian priors in the setting of Garg et al. (2022). A complementary approach to explaining ICL has focused on explicit weight constructions that implement functionalities observed to facilitate ICL, such as gradient descent on the context for linear regression (Li et al., 2024; 2023; Ahn et al., 2023; von Oswald et al., 2022; Fu et al., 2024) and statistical induction heads for Markov chains (Edelman et al., 2024; Rajaraman et al., 2024; Chen et al., 2024). However, relatively few studies have investigated how—or whether—these weight configurations can actually be reached during training (Zhang et al., 2023; 2024). None of these theoretical frameworks has addressed the specific setting of multiple task categories with hierarchical complexity relationships. In this work, we extend the Bayesian viewpoint to explain how transformers select the simplest sufficient hypothesis in both linear regression and Markov chain settings. While these two domains differ in several respects—the format of the prompt (pairs of inputs vs. stream of symbols), the optimization objective (squared loss vs. next-token prediction), and the underlying mechanisms (gradient descent vs. statistical induction heads)—we investigate both settings under a unifying prism.

Algorithm Selection. Most closely related to our work are Lin & Lee (2024); Bai et al. (2023) investigating ICL regression across multiple hypotheses. Lin & Lee (2024) considers ICL of linear regression where features $\mathbf{x}$ and tasks $\mathbf{w}$ are drawn from Gaussian mixtures, generalizing the setting of Garg et al. (2022). This richer framework enables a principled study of the task retrieval versus learning modes through a Bayesian perspective. Similarly, Bai et al. (2023) investigates ICL under tasks from multiple categories, such as linear regression, noisy linear regression, and linear classification. They demonstrate that transformers implement algorithm selection in context, applying the most appropriate algorithm for the provided context at inference time. However, these works do not address tasks from different categories with clearly defined hierarchical complexity relationships, where a higher-complexity category can fully explain the context generated by simpler categories. In this particularly challenging setting, we show that algorithm selection is implemented via a form of Bayesian Occam's razor, allowing transformers to identify and apply the simplest sufficient hypothesis. We demonstrate this both for regression and Markov-chain settings. More recent work by Elmoznino et al. (2025) posits that a meta objective of ICL is linked to a prequential coding algorithm that simultaneously minimizes both algorithm complexity and prediction error on in-context demonstrations. While this intriguing interpretation reveals a form of simplicity preference in ICL, it differs fundamentally from our focus on hypothesis selection across explicitly defined hierarchical complexity structures.

B. Additional Details for Section 2

B.1. Markov chains: Omitted training details

The train loss is written as

$$L(\theta) := \mathbb{E}_{\substack{X\sim P_s\sim \mathrm{Dir}(\mathbf{1})^{\otimes V^s} \\ s\sim \mathrm{Unif(ord)}}} \left[\sum\nolimits_{t=1}^{T} \ell(M_\theta(X_{\le t}), x_{t+1}) \right]. \tag{7}$$

Order-k statistics are computed as

$$\sum_{t=k}^{T-1} \frac{\mathbb{1}(x_{t-1}=x_T, x_{t-2}=x_{T-1}, ..., x_{t-k+1}=x_{T-k}, x_t=v)}{\sum_{t=k}^{T} \mathbb{1}(x_{t-1}=x_T, x_{t-2}=x_{T-1}, ..., x_{t-k+1}=x_{T-k})}, \; v \in [V]. \tag{8}$$

B.2. Markov chains: Omitted details for asymptotics of likelihood

Using a BIC style Laplace approximation to the marginal likelihood (Schwarz, 1978; Ghosal & van der Vaart, 2017; Csiszár & Talata, 2006), we have

$$\log p(X \mid s) \;\approx\; \log p\big(X \mid \hat{P}_s\big) \;-\; \tfrac{d_s}{2} \log T \tag{9}$$

Here, $\hat{P}_s$ denotes the parameters of the order-s chain that maximize the likelihood of the observed sequence X. d_s is the effective dimension of an order s chain, concretely, $d_s = V^s(V-1)$. For an order-s chain, we know that

$p(X \mid P_s) = \prod_t p_{P_s}(x_t \mid x_{t-1}, ..., x_{t-s})$, and the parameters $\hat{P}_s$ that maximize this are the empirical conditional probabilities derived from the observed sequence, i.e. $p_{P_s}(x_t \mid x_{t-1}, ..., x_{t-s}) = \hat{p}(x_t \mid x_{t-1}, ..., x_{t-s})$ for each context of length s. Formally, $\hat{p}_X(x_t \mid x_{t-1}, ..., x_{t-s}) = \frac{n_{x_{t-s}, ..., x_{t-1}, x_t}(X)}{n_{x_{t-s}, ..., x_{t-1}}(X)}$, where $n_{x_i, ..., x_j}(X)$ denotes the number of symbol transitions $x_i \to \ldots \to x_j$ in the sequence X.

Saturation to the Higher Order. Now, suppose that the true process is of order-k, rather than (say) order-1, where $k > 1$. The model order complexity term in Eqn. (9) always favors the lower order (here $= 1$). However, in this case the empirical likelihood terms differ to each other when evaluated for $s = k$ vs $s = 1$. Previous works (Ghosal & van der Vaart, 2017; Csiszár & Talata, 2006) have shown that the *per-sample* improvement in log likelihood under the correct, higher-order model accumulates linearly in T. Concretely, for large T and some universal constant $c > 0$,

$$\log p(X \mid s = k) - \log p(X \mid s = 1) \approx cT - \tfrac{(V^{k-1}-1)V(V-1)}{2} \log T \overset{T \gg k}{\Longrightarrow} p(s = k \mid X) \approx 1.$$

Thus, the posterior probability correctly *saturates* to the true higher order once T is large.

B.3. Linear Regression: Omitted training details

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 $\mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_s)$. For task dimension $s = d/2$, we effectively use $\mathbf{w} = [\mathbf{w}_s, \mathbf{0}]$ (zero-padded to dimension d). We then sample feature vectors $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ and generate the labels $y_t = \mathbf{w}^\top \mathbf{x}_t$ for $t \in [T]$, forming sequences $X_{\le t} = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_t, y_t]$. We use this to train a transformer M_θ to auto-regressively predict the label y_{t+1} using the first t pairs of examples $X_{\le t}$ by minimizing the population square-loss (analogous to Eqn. (6)).

$$L(\theta) := \mathop{\mathbb{E}}_{\substack{\mathbf{x}_{1:T} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d) \\ \mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_s) \\ s \sim \text{Unif(dim)}}} \sum_{t=1}^{T} \ell(M_\theta(X_{\le t}; \mathbf{x}_{\text{test}}), y_{t+1}), \tag{10}$$

where ℓ is the squared loss.

B.4. Bayesian Interpretation of the Selection Mechanism for Linear Regression

The Bayes' optimal prediction on the test query $\mathbf{x}_{\text{test}}$ given the in-context demonstrations is $y = \mathbf{w}_{\text{BO}}^\top \mathbf{x}_{\text{test}}$, where $\mathbf{w}_{\text{BO}}$ is the mixture: $\mathbf{w}_{\text{BO}} = \sum_{d' \in \{d/2,\, d\}} p(d' \mid A_d, \mathbf{y}) \cdot \tilde{\mathbf{w}}_{d'}^{\text{LS}}$. Here, $p(d' \mid A_d, \mathbf{y})$ denotes the posterior probability of the d'-dimensional regressor given the matrix vector A_d and the labels $\mathbf{y}$, and $\tilde{\mathbf{w}}_{d'}^{\text{LS}} = [\mathbf{w}_{d'}^{\text{LS}}, \mathbf{0}]$. Using Bayes' theorem, we express the posterior probabilities in terms of the likelihoods (assuming a uniform prior on regressor dimensionality): $p(d' \mid A_d, \mathbf{y}) = p(\mathbf{y} \mid A_d, d') \big/ \sum_{r \in \{d, d/2\}} p(\mathbf{y} \mid A_d, r)$. For Gaussian priors over $\mathbf{w}_{d'}$, we can derive analytical expressions for these likelihoods. Similar to our Markov chain analysis, the likelihood incorporates a complexity penalty that favors lower-dimensional models. Consequently, when the underlying regressor generating sequence X is $\mathbf{w}^* = \mathbf{w}_{d/2}$ and $d > T > d/2$, we find that $p(d/2 \mid A_d, \mathbf{y}) \gg p(d | A_d, \mathbf{y})$, resulting in $\mathbf{w}_{\text{BO}} \approx \mathbf{w}_{d/2}^{\text{LS}}$. Conversely, when $\mathbf{w}^* = \mathbf{w}_d$, the improved fit outweighs the complexity penalty, giving us $\mathbf{w}_{\text{BO}} = \mathbf{w}_d^{\text{LS}}$. This Bayesian mechanism explains why transformers implement Occam's razor by selecting the simplest hypothesis in-context.

B.5. Bayes' Optimal for Linear Regression

In this section, we characterize the Bayes' optimal parameter $\mathbf{w}_{\text{BO}}$ under the mixture of regressors described in the section above. For square loss, the Bayes optimal parameter is the posterior mean of the $\mathbf{w}$ conditioned on the sequence X, i.e. $\mathbf{w}_{\text{BO}} = \mathbb{E}[\mathbf{w} \mid A_d, \mathbf{y}]$, where A_d is the matrix of vectors and $\mathbf{y}$ is the vector of labels derived from the sequence X.

In order to find this posterior mean, we need the conditional distribution $p(\mathbf{w} \mid A_d, \mathbf{y})$. By Bayes' rule,

$$p(\mathbf{w} \mid A_d, \mathbf{y}) = \frac{p(\mathbf{y} \mid A_d, \mathbf{w})\, p(\mathbf{w})}{\int p(\mathbf{y} \mid X, \mathbf{w}')\, p(\mathbf{w}')\, \mathrm{d}\mathbf{w}'}. \tag{11}$$

Under uniform prior on the regressor dimensionality, the overall prior $p(\mathbf{w}) = \frac{1}{2} p_{d/2}(\mathbf{w}) + \frac{1}{2} p_d(\mathbf{w})$, where $p_{d/2}(\mathbf{w})$, $p_d(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ denote the respective priors over the $d/2$ and d dimensional regressors. Here, $p_{d/2}(\mathbf{w})$corresponds to a

Gaussian $\mathcal{N}(\mathbf{0}, \mathbb{I}_{d/2})$ over the first $d/2$ components of $\mathbf{w}$ with zeroes in the remaining $d/2$. Plugging this into Equation (11), and using the fact the $\mathbf{w}_{d/2}^{\text{LS}}$ and $\mathbf{w}_{d}^{\text{LS}}$ are the Bayes' optimal solutions under their respective priors, we get a mixture of solutions

$$\mathbb{E}[\mathbf{w} \mid A_d, \mathbf{y}] = \frac{L_{d/2}}{L_{d/2} + L_d} \tilde{\mathbf{w}}_{d/2}^{\text{LS}} + \frac{L_d}{L_{d/2} + L_d} \tilde{\mathbf{w}}_{d}^{\text{LS}},$$

where $L_{d'} = p(\mathbf{y}|A_d, d')$ denote the marginal likelihood of $\mathbf{y}$ under the regressor of dimensionality d'. Notice the term $\frac{L_{d'}}{\sum_r L_r} = p(d'|A_d, \mathbf{y})$ using Bayes' rule and uniform prior over the dimensionality of regressor. Thus, the Bayes' optimal is the following posterior-weighted combination of the respective LS solutions

$$\mathbf{w}_{\text{BO}} = \sum_{d' \in \{d/2,\ d\}} p(d' \mid A_d, \mathbf{y}) \cdot \tilde{\mathbf{w}}_{d'}^{\text{LS}}.$$

Expressions for L_1 and L_2. The marginal likelihood $L_{d'}$ under the noiseless linear regression setting is

$$L_{d'} \;=\; \int \delta\big(\mathbf{y} - A_d\,\mathbf{w}\big)\, p_{d'}(\mathbf{w})\, \mathrm{d}\mathbf{w}.$$

This can be interpreted as the "density" of $\mathbf{y}$ when viewed as a random variable $A_d\,\mathbf{w}$ for $\mathbf{w} \sim p_{d'}$. Concretely, if the sequence generating regressor $\mathbf{w}^* \sim p_d(\mathbf{w})$, then $L_{d/2} = 0$ (a.s.), as no $\mathbf{w} \sim p_{d/2}$ solves $\mathbf{y} = A_d\mathbf{w}$. Hence, $\mathbf{w}_{\text{BO}} = \mathbf{w}_d^{\text{LS}}$.

Bayesian Occam's Razor. Now consider the other case when $\mathbf{w}^* \sim p_{d/2}(\mathbf{w})$. Here, regressors from either prior can "explain" the context and will have non-zero marginal likelihood. Consider the regime $d > T \geq d/2$, the density of $\mathbf{y}$ under $p_{d/2}$ is

$$\begin{aligned} L_{d/2} \;&=\; \frac{1}{(2\pi)^{d/4}\sqrt{\det(A_{d/2}^\top A_{d/2})}} \exp\Big(-\tfrac{1}{2}\left\|\mathbf{w}_{d/2}^{\text{LS}}\right\|^2\Big), \\ L_{d} \;&=\; \frac{1}{(2\pi)^{T/2}\sqrt{\det(A_{d}\, A_{d}^\top)}} \exp\Big(-\tfrac{1}{2}\left\|\mathbf{w}_{d}^{\text{LS}}\right\|^2\Big). \end{aligned}$$

As A_d has i.i.d. Gaussian entries, both $A_d A_d^\top$ and $A_{d/2}^\top A_{d/2}$ follow Wishart distributions. It is thus known (e.g., (Tulino et al., 2004)) that

$$\begin{aligned} \mathbb{E}\log\det(A_d A_d^T) &= \psi_T(d/2) + T\log 2, \\ \mathbb{E}\log\det(A_{d/2}^T A_{d/2}) &= \psi_{d/2}(T/2) + (d/2)\log 2, \end{aligned}$$

where ψ_p is the multivariate digamma function. It further holds that

$$\psi_T(d/2) - \psi_{d/2}(T/2) = \sum_{i=1}^{T} \psi\left(\frac{d+1-i}{2}\right) - \sum_{j=1}^{d/2} \psi\left(\frac{T+1-j}{2}\right),$$

where ψ is the digamma function. Using asymptotic expansions of the digamma function, an estimate for this difference when T and d grow proportionally large is (Cai et al., 2015) $\psi_T(d/2) - \psi_{d/2}(T/2) \approx \frac{d}{2}\log\frac{T}{d/2}$. In fact, (Cai et al., 2015) shows that in this regime, the log-determinants concentrate around their expectations shown above. Thus,

$$\frac{\det(A_d A_d^T)}{\det(A_{d/2}^T A_{d/2})} \approx \left(\frac{T}{d/2}\right)^{d/2}.$$

At the same time, it is easy to show that $\|\mathbf{w}_d^{\text{LS}}\|^2 - \|\mathbf{w}_{d/2}^{\text{LS}}\|^2 > 0$. Together, $L_{d/2}/L_d \gtrsim (2\pi)^{T/2-d/4}\left(\frac{T}{d/2}\right)^{d/2}$, which is $\gg 1$ for large d and $T > d/2$. This implies that, $p(d/2 \mid A_d, \mathbf{y}) \approx 1$, and $\mathbf{w}_{\text{BO}} \approx \mathbf{w}_{d/2}^{\text{LS}}$.

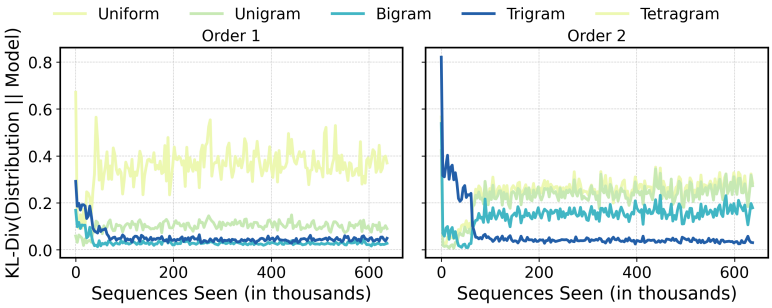

Figure 3. A transformer trained on sequences generated by random order-1 and order-2 Markov chains can infer whether the context is order-1 or order-2, then generate predictions using the corresponding bigram or trigram statistics. **(Left)** shows the distance between the model's output distribution 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.

C. Additional Results

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. Figure 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 Figure 5) and observe a similar behaviour.

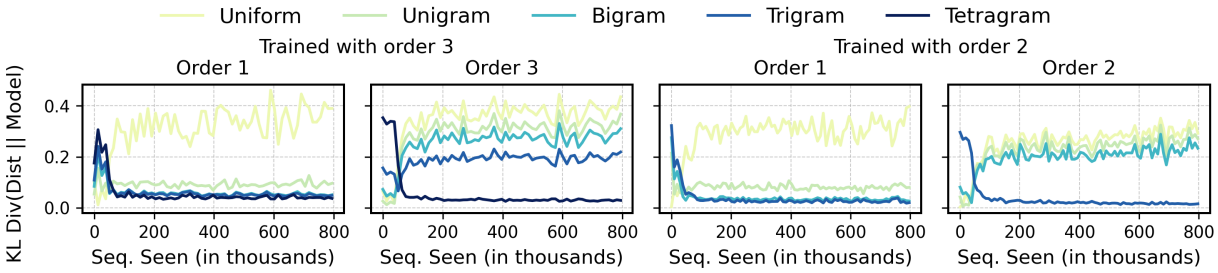

Figure 4. A transformer trained on random order-3 (left column) or order-2 (right column) Markov chains can only predict based on order-3 (left column) or order-2 (right column) statistics, and fails to predict based on order-1 statistics when given order-1 in-context sequences.

Testing with different context lengths. In Figures 6 and 7, we test how varying the context length affects a trained transformer's behaviour on sequences from different task categories in the Markov chain setting and the PCFG setting, respectively.

Evaluation results with training time for linear regression. Figure 8 shows the how the predictions of the transformer evolve with training time compared to the two benchmark LS solutions.

C.1. Probabilistic Grammars

Several recent works (Allen-Zhu & Li, 2024; Liu et al., 2023a;b; Ahuja et al., 2025) use formal languages to understand language modeling. In this section, we study probabilistic context-free grammars (PCFGs), which are stochastic systems designed to generate sequences of symbols from specified vocabularies (see **??** for a formal description), to verify our conclusions about ICL of tasks of varying complexity.

In our setup, non-terminal symbols $\texttt{N} := \{\texttt{S}, \texttt{A}, \texttt{B}, \texttt{C}\}$, terminal symbols $\texttt{T} := \{\texttt{a}, \texttt{b}, \texttt{eos}\}$, where $\texttt{eos}$ corresponds to the end-of-string symbol. The 'complex' PCFGs use the following set of probabilistic productions:

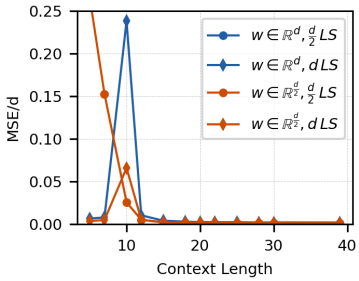

Figure 5. A transformer trained on $X = [\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, ..., \mathbf{x}_T, y_t]$ sequences generated using fixed complexity, random d-dimensional regressors $\mathbf{w}_d$ (see Section 2 for details). We plot $(M_\theta([X, \mathbf{x}_{\text{test}}]) - \mathbf{w}_{\text{bench}}^\top \mathbf{x}_{\text{test}})^2/d$ where $\mathbf{w}_{\text{bench}}$ refers to two benchmark least-squares solutions in d or $d/2$ dimensional space described in Section 2. 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 Figure 2. Here, $T = 39$ and $d = 10$. Also the first curve (blue, dot) is out of the plotted range.

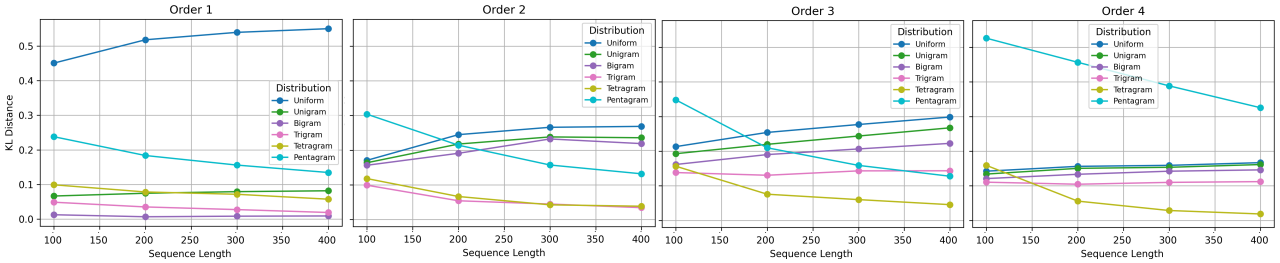

Figure 6. Figure illustrates how varying the context length affects a trained transformer's performance when prompted with sequences from different-order chains at inference. This transformer was trained on sequences from order-1 and order-3 chains (Figure 1). Here, we also evaluate its performance on 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 order-4 sequences; it applies the strategy that best explains the observed context at inference time.

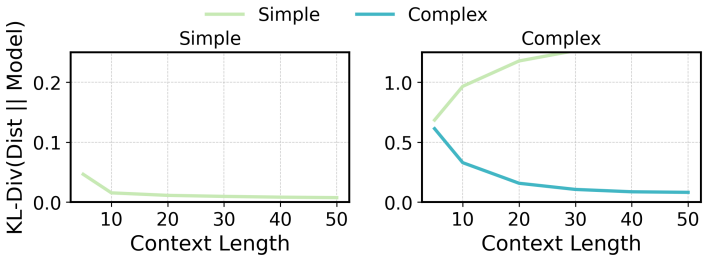

Figure 7. Figure illustrates the performance of a trained transformer when prompted with sequences of varying context lengths from the two grammars. The context length during training was 50 in this case. We find that the transformer can identify the correct type of grammar for slightly shorter context lengths, but the performance starts deteriorating for very short context lengths.

$$
\begin{aligned}
&\texttt{S} \to \texttt{ABC} \mid \texttt{BAC} \mid \texttt{AAC} \mid \texttt{BBC} && \text{with probabilities } p_1, p_2, p_3, p_4, \\
&\texttt{A} \to \texttt{a}, \quad \texttt{B} \to \texttt{b} && \\
&\texttt{C} \to \texttt{S} \mid \texttt{eos} && \text{with probabilities } q_1, q_2.
\end{aligned} \tag{12}
$$

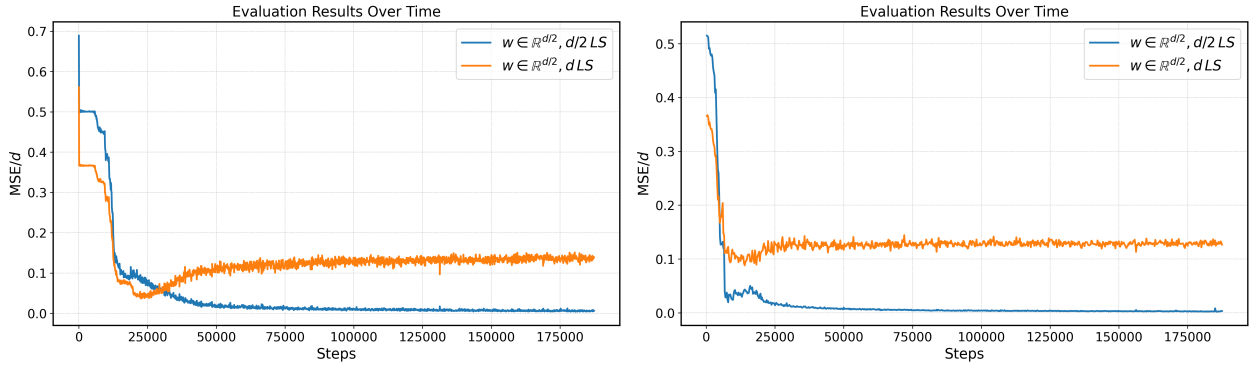

Figure 8. Figure illustrates the inference results across train time for $d = 10$ (**left**) and $d = 20$ (**right**) using contexts of fixed length $T = 5$ and $T = 15$, respectively. In both cases, the transformer's predictions eventually match (closely) to the $\mathbf{w}^{\text{LS}}_{d/2}$.

The 'simple' PCFGs are a subset of the 'complex' PCFGs, where we enforce $p_3 = p_4 = 0$.

To generate sequences from this set of PCFGs, we first draw the probabilities p_i and q_i randomly from a uniform Dirichlet distribution. Then, we keep generating strings from this PCFG (starting from `S` and following the production rules, till it returns `eos`) and concatenating them till the total sequence length matches the context length T. To control the complexity of generated sequences and prevent infinite recursion, we impose a maximum derivation depth $d_{\max}$; once this depth is reached, expansions of the nonterminal `C` are restricted to produce the terminal symbol `eos`. For generating training sequences, we set $d_{\max} = 10$, whereas for test sequences, it is set as 0, *i.e.*, we only get strings from the set {`abeos, baeos, aaeos, bbeos`} with different probabilities.

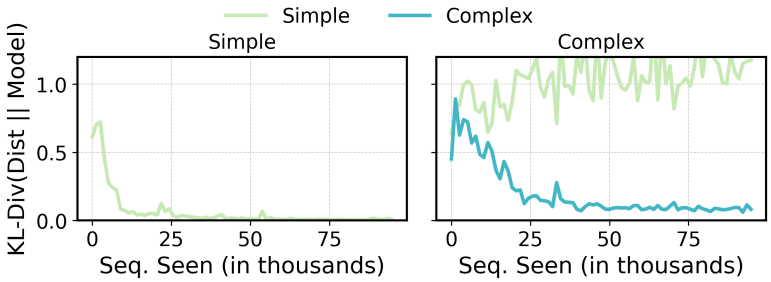

Figure 9. A transformer trained on sequences from random 'complex' and 'simple' PCFGs (see Equation (12)) can infer the type of PCFG the context comes from. Here, the test sequences are of the form $\{\ldots, \mathtt{a}\}$ or $\{\ldots, \mathtt{b}\}$, where every string before the last element is from the set {`abeos, baeos, aaeos, bbeos`}. **(Left)** compares the KL divergence between model output probabilities and the probabilities of the 'simple' PCFG. **(Right)** compares the KL divergence between model output probabilities and the probabilities of the ground truth 'complex' PCFG as well as those of the 'simple' PCFG.

In Figure 9, we train a transformer on sequences drawn from the two types of PCFGs. The test sequences are of the form $\{\ldots \mathtt{eos}, \mathtt{a}\}$ or $\{\ldots \mathtt{eos}, \mathtt{b}\}$. For sequences from the 'complex' grammar, we measure the KL divergence between the transformer's output distribution with the ground truth probabilities of the grammar, as well as those of the 'simple' PCFG. Note that since $p_3 = p_4 = 0$ for the 'simple' PCFG, $P(\mathtt{a}|\mathtt{a}) = P(\mathtt{b}|\mathtt{b}) = 0$. We see that the transformer output distribution is closer to the 'complex' PCFG. For sequences from the 'simple' grammar, we find that the output distribution closely matches the 'simple' PCFG. This result shows that the transformer infers the type and parameters of the PCFG from which the context was generated. Here, we use the same context length for train and test sequences; please see Appendix C for test results with different context lengths.

D. Construction for Markov chains of varying complexity

Recall from Eqn. (1) that the model learns tasks of different complexity by implementing the Bayes optimal strategy. To achieve this, the model must compute several key quantities, including the s-gram statistics $p(\cdot \mid x_T, \ldots, x_{T-s+1})$ and the

category posterior.

Previous work has shown that a two-layer attention-only model with s heads in the first layer can compute the relevant s-gram statistics for the last token (Edelman et al., 2024; Rajaraman et al., 2024). To extend those results, for order one statistics, we demonstrate that by using V heads in the second layer, the model can represent all the conditional distributions $p(u \mid v)$ for any pair of vocabulary tokens simultaneously. These conditional distributions are the building blocks that produce the log-likelihood used to compute the posterior. This approach can be extended to higher-order Markov chains by using V^s heads in the second layer.

Lemma D.1. *A two-layer attention-only transformer with one attention head in the first layer, and V attention heads in the second layer can output* all *the empirical conditional distributions*

$$p(u \mid v) = \frac{\sum_{i=2}^{T} \mathbb{1}(x_{i-1} = v, x_i = u)}{\sum_{i=2}^{T} \mathbb{1}(x_{i-1} = v)} \quad \textit{for any } u, v \in V.$$

for any input sequence X.

Proof. We outline how to set the transformer's parameters so that, for each pair (v, u), the model's final output can encode the conditional probability $p(u \mid v)$.

Recall that the input to the model is a sequence $X = [x_1, \cdots, c_T]$ of length T with $x_t \in [V]$. Let $z_t \in \mathbb{R}^d$ be the input embedding of the t-th element of the sequence x_t, and $Z = [z_1, z_2, \ldots, z_T]^\top_{T \times d}$ be the sequence of input embeddings.

We write the two-layer transformer with V heads in the second layer as

$$f(X) \;=\; W_o \big[\, h_1(Z) \,\|\, \cdots \| \, h_V(Z) \big],$$

where "$\|$" denotes concatenation, $W_o \in \mathbb{R}^{V^2 \times dV}$ is the final linear projection, and each $h_v(Z)$ is the output of the v-th attention head in the second layer. Specifically,

$$h_v(Z) \;:=\; (\mathrm{Attn}_2^v + I) \circ (\mathrm{Attn}_1^v + I)\,(Z).$$

with

$$\mathrm{Attn}(Z) \;=\; \mathrm{softmax}\big(\mathrm{mask}(A)\big)\, Z W_V,$$

where W_V is the value matrix, and the attention map $A \in \mathbb{R}^{T \times T}$ is defined as

$$A_{i,j} \;=\; \frac{(z_i^\top W_Q \;+\; r_{i-j+1}^\top)\, W_K\, z_j^\top}{\sqrt{d}}.$$

Here W_Q, W_K, are the key and query matrix, r_k is the relative positional embedding, and $\mathrm{mask}(\cdot)$ enforces causal masking so that position i can only attend to positions $1, \ldots, i$.

We will show how to choose the model's weights so that, for each v, the embedding from the v-th head of the second layer, $h_v(Z)$, produces vectors which—when multiplied by W_o—yield the conditional distribution $p(\cdot \mid v)$ across the sequence X. For simplicity, we will omit the superscript v in Attn_ℓ^v from now on.

Layer 1: Isolating the previous token. We first show how to configure the weights of the first layer so that each position i copies the immediately preceding token z_{i-1}. For this construction we need the embedding dimension of $d = 3V$.

Set the embedding matrix $E = [I_V, I_V, 0]_{V \times d}$. In this case, the input embedding z_t is $= [\underbrace{z_t^{1\top}}_{V}, \underbrace{z_t^{2\top}}_{V}, \underbrace{z_t^{3\top}}_{V}]^\top =$ $[e_{x_t}^\top, e_{x_t}^\top, 0]^\top$ where e_v is the v-th basis vector.

Now let the key, query matrix and the positional embeddings be as follows.

$$W_Q = 0, \quad W_K \;=\; \begin{pmatrix} c\, I_{V \times V} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad R \;=\; e_2 \cdot 1^\top,$$

where c is a large constant. With these weights, the attention matrix A is zero for all entries except when $j = i - 1$; that is, $A_{i,j} = 0$ if $j \neq i - 1$. For the entry where $j = i - 1$, we have $A_{i,i-1} = 1^\top W_K z_{i-1} = c$. As c approaches infinity, the softmax function saturates. In other words, if we define

$$S = \mathrm{softmax}\big(\mathrm{mask}(A)\big),$$

then $S_{i,j} = 1$ when $j = i - 1$ and 0 for all other j. This ensures that the model only attends to the previous position in the first layer.

Now choosing

$$W_V \;=\; \begin{pmatrix} 0 & 0 & 0 \\ 0 & I_{V\times V} & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

we have

$$\mathrm{Attn}_1(Z)_{i,:} = \sum_{t=1}^{T} S_{i,t} z_t^\top W_V = [0, {z_{i-1}^2}^\top, 0]^\top.$$

This extracts and carries along the "middle" V components of the embedding at position $i - 1$. Putting things together, the embedding of after the first layer will be

$$z_i' \;:=\; \left(\mathrm{Attn}_1 + I\right)(Z)_i = \left[\, z_i^1,\; z_i^2 + z_{i-1}^2,\; z_i^3 \right].$$

Layer 2 - v-th head: Computing conditional probabilities $p(\cdot \mid v)$. We now use V heads in the second layer to encode, for each possible previous token $v \in V$, how likely the next token is $u \in V$.

In this layer, we set the relative positional embedding $r = 0$ and

$$W_Q \;=\; \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1_v\, e_v^\top & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad W_K \;=\; c \begin{pmatrix} 0 & 0 & 0 \\ -e_v\, e_v^\top & e_v\, e_v^\top & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

With this weight configuration, we have

$$A_{T,i} = {z_T'}^\top W_Q W_k^\top z_i' = \begin{cases} c & \text{if } x_{i-1} = v \\ 0 & \text{o.w.} \end{cases}$$

Then, as $c \to \infty$,

$$S_{T,i} = \frac{\mathbb{1}(x_{i-1} = v)}{\sum_{t=1}^{T} \mathbb{1}(x_{t-1} = v)}$$

Now setting

$$W_V \;=\; \begin{pmatrix} 0 & 0 & I_V \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

we have

$$\begin{aligned} h_v(Z) := \sum_{t=1}^{T} S_{T,t} W_V^\top z_t' + z_T' &= \sum_{u\in[V]} \frac{\sum \mathbb{1}(x_{i-1} = v, x_i = u)}{\sum \mathbb{1}(x_{i-1} = v)} [0, 0, e_u^\top]^\top + z_T' \\ &= \sum_{u\in[V]} p(u \mid v)[0, 0, e_u^\top]^\top + z_T' \end{aligned}$$

Finally, with $P_{V\times 3V} = [0, 0, I]$, setting W_O as

$$W_O = \begin{pmatrix} P & 0 & \cdots & 0 \\ 0 & P & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & P \end{pmatrix},$$

we have

$$\begin{aligned} f(Z) &= W_O[\, h_1(Z) \parallel \cdots \parallel h_V(Z)] \\ &= [\sum_u p(\cdot \mid v = 1)e_u \parallel \cdots \parallel \sum_u p(\cdot \mid v = V)e_u] \end{aligned}$$

□

E. 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 $1e-4$. We set the batch size as 32.

Markov chains. For all Markov chain experiments, vocab size $V = 3$. For the order-1 and order-3 experiments in Figure 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 3, context length T was set to 200. We used a 2 layer transformer with 20 heads, and embedding dimension was set to 320. Additionally, we used relative position encoding in all the experiments. For the fixed-order experiments in Figure 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). We set $d = 10$ and 20 (for Figure 2), and context length $T = 39$.

PCFGs. We used a 4 layer, 4 head transformer with embedding dimension 128. We use context length of 50.

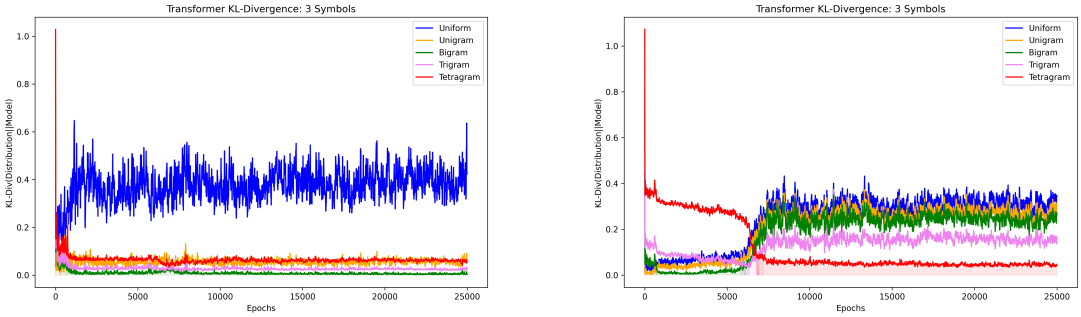

Figure 10. Repetition of the experiment in Figure 1 when training with context length $T = 400$ on a mixture of order-1 and order-3 chains.

F. Ablations

In this section, we experiment with different context lengths and model sizes for the Markov chain setup. In Figures 10 to 14, every epoch refers to an iteration with 32 (batch size) number of sequences.

Figure 10 shows the results for training on mixture of order-1 and order-3 chains with context length 400. The results are consistent with those in Figure 1.

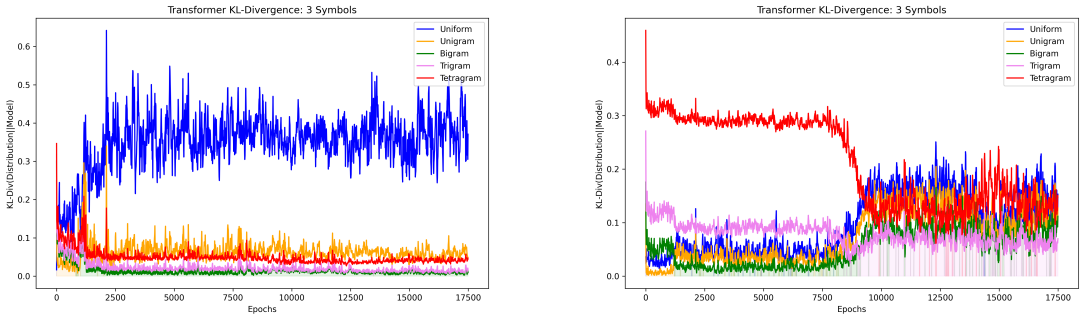

Figure 11. Repetition of the experiment in Figure 1 when training on a mixture of order-1 and order-3 chains with a 4 layer, 4 head architecture, and embedding dimension 96.

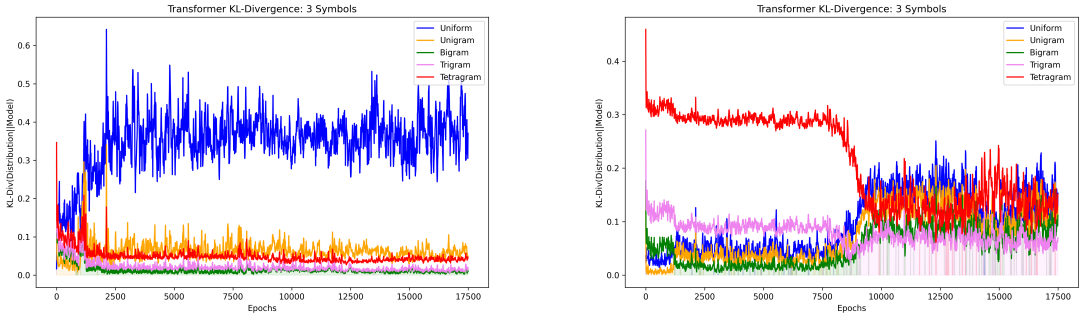

Figure 12. Repetition of the experiment in Figure 1 when training on a mixture of order-1 and order-3 chains with a 4 layer, 4 head architecture, and embedding dimension 96. Training is done with varying context length per order, with $T_s = 100 * s$, where T_s denotes the context length for sequence X generated by a chain of order-s. Inference is conducted using a fixed $T = 300$.

Smaller architectures. In Figures 11 and 12, we repeat the experiment in Figure 1 on a smaller transformer model with 4 layers, 4 heads and embedding dimension 96. In Figure 11, the context length is 300 for all sequences, while in Figure 12, we consider context lengths 100 and 300 for sequences from order-1 and order-3 chains, respectively. We find that the smaller model is unable to learn tetragram statistics on order-3 chains and instead switches between bigram and trigram statistics depending on the order.

2 layer architectures. In Figures 13 and 14, we repeat the experiments in Figures 1 and 3, respectively with a 2 layer transformer model, varying the embedding dimension and number of heads in the first layer, while keeping the heads in the second layer fixed to 1. We find that across all settings, the transformer infers the correct order from the context. We also observe that scaling the model size speeds up convergence.

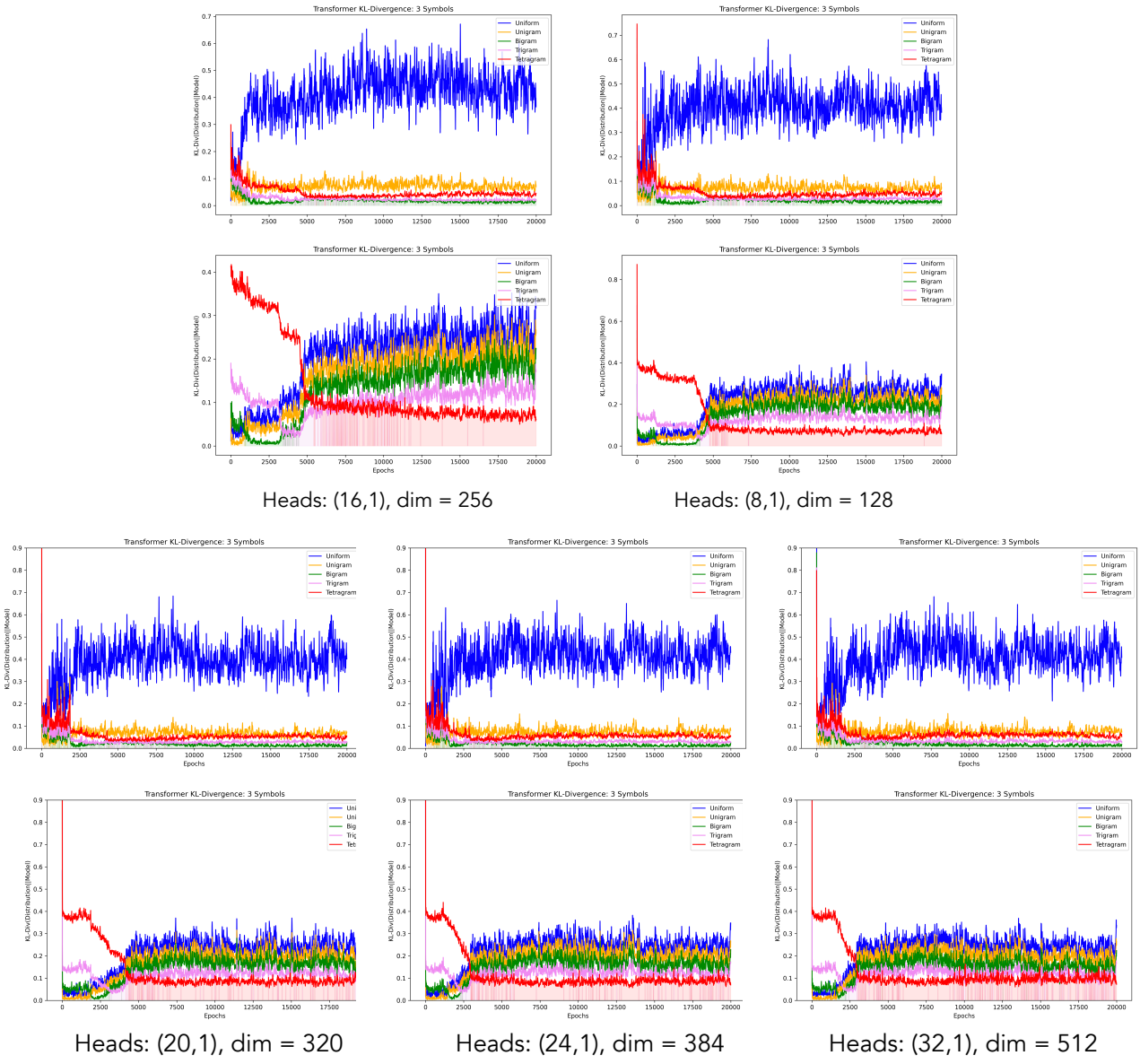

Figure 13. Repetition of the experiment in Figure 1 when training on a mixture of order-1 and order-3 chains with a 2 layer transformer, with n heads in the first layer, 1 head in the second layer, and embedding dimension *dim*.

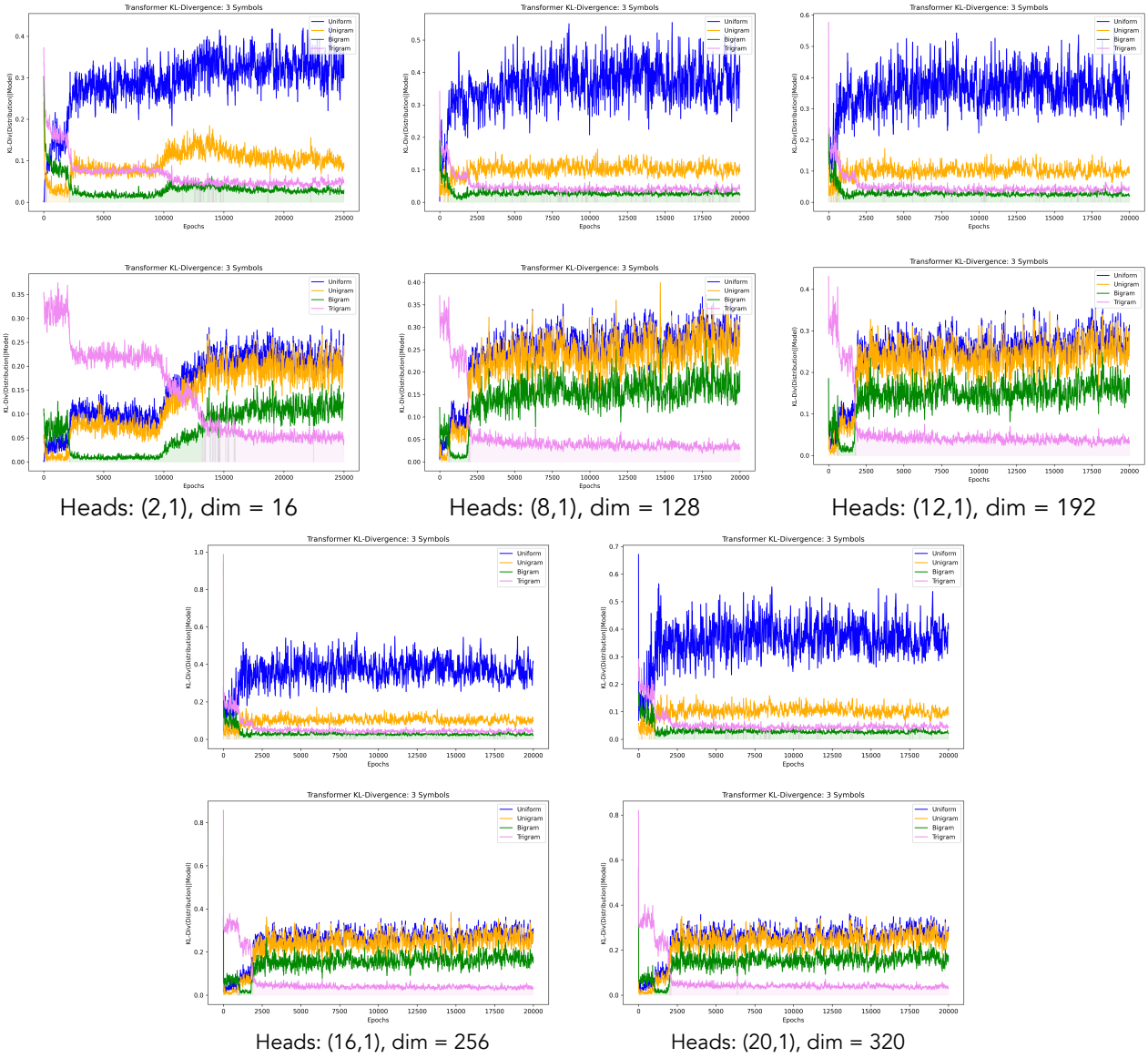

Figure 14. Repetition of the experiment in Figure 3 when training on a mixture of order-1 and order-2 chains with a 2 layer transformer, with n heads in the first layer, 1 head in the second layer, and embedding dimension *dim*.