GENOMIC NEXT-TOKEN PREDICTORS ARE IN-CONTEXT LEARNERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) – the capacity of a model to infer and apply abstract patterns from examples provided within its input – has been extensively studied in large language models trained for next-token prediction on human text. In fact, prior work often attributes this emergent behavior to distinctive statistical properties in *human* language. This raises a fundamental question: can ICL arise *organically* in other sequence domains purely through large-scale predictive training?

To explore this, we turn to genomic sequences, an alternative symbolic domain rich in statistical structure. Specifically, we study the Evo2 genomic model, trained predominantly on next-nucleotide (A/T/C/G) prediction, at a scale comparable to mid-sized LLMs. We develop a controlled experimental framework comprising symbolic reasoning tasks instantiated in both linguistic and genomic forms, enabling direct comparison of ICL across genomic and linguistic models. Our results show that genomic models, like their linguistic counterparts, exhibit log-linear gains in pattern induction as the number of in-context demonstrations increases. To the best of our knowledge, this is the first evidence of organically emergent ICL in genomic sequences, supporting the hypothesis that ICL arises as a consequence of large-scale predictive modeling over rich data. These findings extend emergent meta-learning beyond language, pointing toward a unified, modality-agnostic view of in-context learning.

1 INTRODUCTION

Scaling Large Language Models (LLMs) has revealed an unexpected and powerful capacity: in-context learning (ICL) (Radford et al., 2018; Brown et al., 2020a), the ability to infer and apply abstract patterns purely from examples contained within their input. Almost all prominent evidence of emergent ICL so far (Srivastava et al., 2023, inter alia) comes from training on human language (e.g., English). This raises a fundamental question: *is there something inherently special about human language that enables ICL to emerge?* To investigate this, we turn to a radically different substrate: **the genome**. Genomic sequences can be viewed as a form of *natural* language that has evolved through nature. Like human language, they comprise complex symbol sequences that exhibit rich statistical regularities – motifs, repeats, dependencies (Benegas et al., 2025) that could, in principle, support pattern induction within context.

Testing for the existence of ICL requires large-scale models. We use the **Evo2** series (Brixi et al., 2025): a large-scale model trained solely on 'next-nucleotide' prediction. With training scale comparable to mid-sized LLMs (e.g., Qwen3-14B (Yang et al., 2025)), these genomic models finally make it possible to systematically compare genomic and linguistic representations under large-scale autoregressive training. If ICL arises primarily from scale and predictive compression, then large-scale genomic models such as Evo2 may already exhibit it.

Our experiment's results (§3) show that large genomic models do, in fact, exhibit ICL. **Both linguistic and genomic models exhibit log-linear improvements** in pattern inference as the number of demonstrations grows. We further validate these findings by demonstrating that Evo2 also exhibits robust few-shot learning on a real-world genomic promoter classification task (§3.1). These findings suggest *ICL is not an artifact that is specific to human language, but a broader consequence of*

LLMs were lightly used during the conception, implementation, and refinement of this paper. We assume full responsibility for all content.

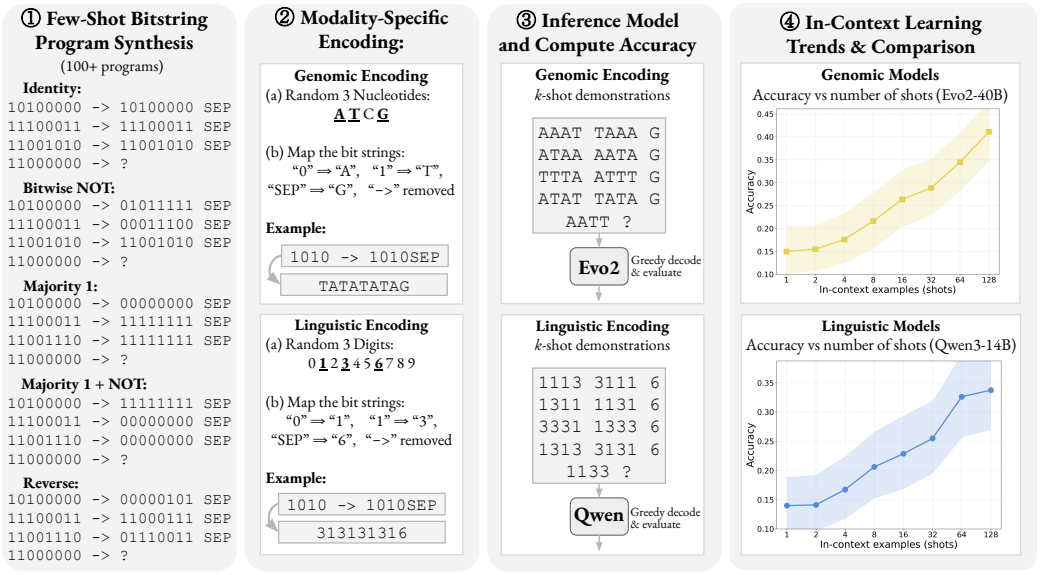


Figure 1: We design parallel symbolic reasoning tasks that allow direct comparison of ICL behavior across modalities. ① Few-shot bitstring program-synthesis tasks require models to infer mappings from examples. ② Each task is rendered in two modality-specific encodings. ③ Both genomic (Evo2) and linguistic (Qwen3) models receive k-shot demonstrations and are greedily decoded to compute exact-match accuracy. ④ **Both models show log-linear accuracy gains with more demonstrations**.

compression and predictive modeling in pattern-rich sequence spaces. This points toward a unified view of context-adaptive computation that spans different modalities of data.

2 Framework for Cross-domain In-context Learning

This section presents our framework for evaluating ICL in linguistic and genomic models. We outline experimental desiderata (§B), setup (§2.1), models (§C), and evaluation protocol (§2.2).

2.1 Experimental Setup

Notation: We define in-context learning (ICL) as follows. Let S and O be the input and output domains, and let each task be a deterministic latent function $f : S \to O$. Given an ordered n-shot context $E = ((x_1, f(x_1)), \ldots, (x_n, f(x_n)))$ with $x_i \in S$ and a query $x \in S \setminus \{x_1, \ldots, x_n\}$, the model predicts $\hat{y} = M(E, x)$. A trial succeeds if $\hat{y} = f(x)$, i.e., $\mathbb{1}[\hat{y} = f(x)]$.

Model families and selection rationale. Details on the model families and our selection criteria are provided in Appendix C.

Task Formulation: in-context program induction over abstract symbols. Given our desiderata (§B), we construct symbolic induction tasks with a small lexicon: the model infers a hidden transformation from a few input–output demonstrations and applies it to a new query. Related setups appear in work on compositional reasoning (Brown et al., 2020b), Raven-style analogical reasoning (Raven et al., 1962; Webb et al., 2023), and ARC-AGI (Chollet, 2019).

Bitstring domain and function space. We use a shared symbolic domain $S = \{0, 1\}^k$ to ensure comparable token granularity across genomic and linguistic models. For genomic models, two nucleotides (e.g., A/C) encode bits while the remaining bases (G/T) serve as separators; for linguistic models, binary digits are typically single tokens (§G; Fig. 1). We set $k = 8$, so $|S| = 2^8 = 256$. Our task set is $F \subseteq \{f : S \to S\}$ with $|F| = 100$, where each f is either one primitive transformation or a composition of two primitives.

Generation of F. We build F from a library of 30 fundamental primitives (listed in §E.1) covering low-level logical operations and higher-order compositional structure. We construct the final function set, we use GPT-5-Codex to automatically generate 100 unique transformations by composing primitives into valid Python programs. The full list of transformations and corresponding source code is provided in §E.2.

2.2 Evaluation Protocol

Evaluation metric. For each transformation $f \in F$, sample i.i.d. trials $(E^{(t)}, x^{(t)})$ with $E^{(t)} \sim \mathrm{Unif}(E_n)$ and $x^{(t)} \sim \mathrm{Unif}(S \setminus E^{(t)})$, where $E_n = \{E \subset S : |E| = n\}$. Given $E^{(t)}$ and $x^{(t)}$, the model predicts $\hat{y}^{(t)} = M(E^{(t)}, x^{(t)})$ and a trial is correct if $\hat{y}^{(t)} = f(x^{(t)})$. Our Monte Carlo estimate of n-shot accuracy is

$$\hat{P}(M, n) = \frac{1}{|F|m} \sum_{f \in F} \sum_{t=1}^{m} \mathbb{1}\left[M(E^{(t)}, x^{(t)}) = f(x^{(t)})\right]. \tag{1}$$

Model suites and sampling. We use $m = 8$ Monte Carlo trials and shot counts $\mathcal{N} = \{1, 2, 4, 8, 16, 32, 64, 128\}$. Genomic models: $\mathcal{M}_G = \{\texttt{evo2\_1b\_base}, \texttt{evo2\_7b}, \texttt{evo2\_40b}\}$; linguistic models: $\mathcal{M}_L = \{\texttt{Qwen3-0.6B-Base}, \texttt{Qwen3-1.7B-Base}, \texttt{Qwen3-4B-Base}, \texttt{Qwen3-8B-Base}, \texttt{Qwen3-14B-Base}\}$. For each $(M, n) \in (\mathcal{M}_L \cup \mathcal{M}_G) \times \mathcal{N}$, we report $\hat{P}(M, n)$. Standard errors are estimated by a two-stage cluster bootstrap (resample f, then its trials) with 5000 replicates per (M, n).

Mode baseline. We define a mode baseline that always predicts the most frequent output observed in the context. We formally define this baseline in §H.

3 Empirical Results

This section reports empirical findings on few-shot bitstring generalization. §3.1 presents the main accuracy trends with respect to model size and shot count. §D.1 examines sensitivity to task complexity using a BitLoad measure, and §D.2 contrasts the models' qualitative competencies across individual transformations.

3.1 Main Results

Across both model families, accuracy generally rises linearly with respect to $\log(\textbf{shots})$. A linear regression linking performance to $\log(\text{shots})$ yields highly significant slopes for all models (all $p \leq 10^{-3}$ via one-sided t-test on slope). As Fig. 3 shows, Evo2 models show cleanly monotonic gains, with a pronounced step from 1B to 7B, and near-indistinguishable curves for 7B and 40B; by 128 shots, both 7B and 40B surpass 40% accuracy (Fig. 3a). Qwen3 also trends upward overall but with non-monotonic patches—especially for smaller models in the 4–16 shot band—before resuming clear improvements from 32 to 128 shots; at 128 shots, the 14B model approaches 35% (Fig. 3b). A full table of all accuracies is in §I.

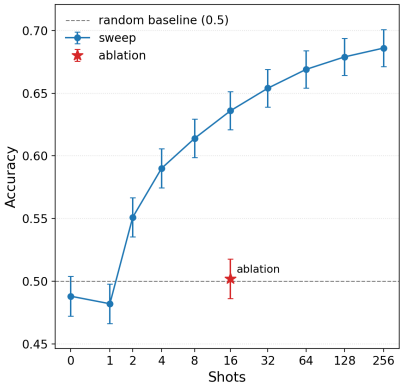


Figure 2: ICL Performance of Evo2 7B on `human_nontata_promoters` at varying shot counts. Error bars are $\pm$ one standard error. The performance shows clear monotonic improvements with respect to shot count.

The mode baseline's performance doesn't improve with more shots, and lags Qwen3 and Evo2 at high shot counts (Fig. 3d). For Qwen3, advantages over the mode baseline become consistently significant ($p < 0.05$ via one-sided z-test on bootstrapped standard errors) at n=128 for all sizes greater than 0.6B. For Evo2, statistically significant advantages emerge slightly earlier: for 1B at n=64, the 7B at $n = 32$, and the 40B at $n = 16$. Regardless, all models surpass the naive baseline at high shot couts.

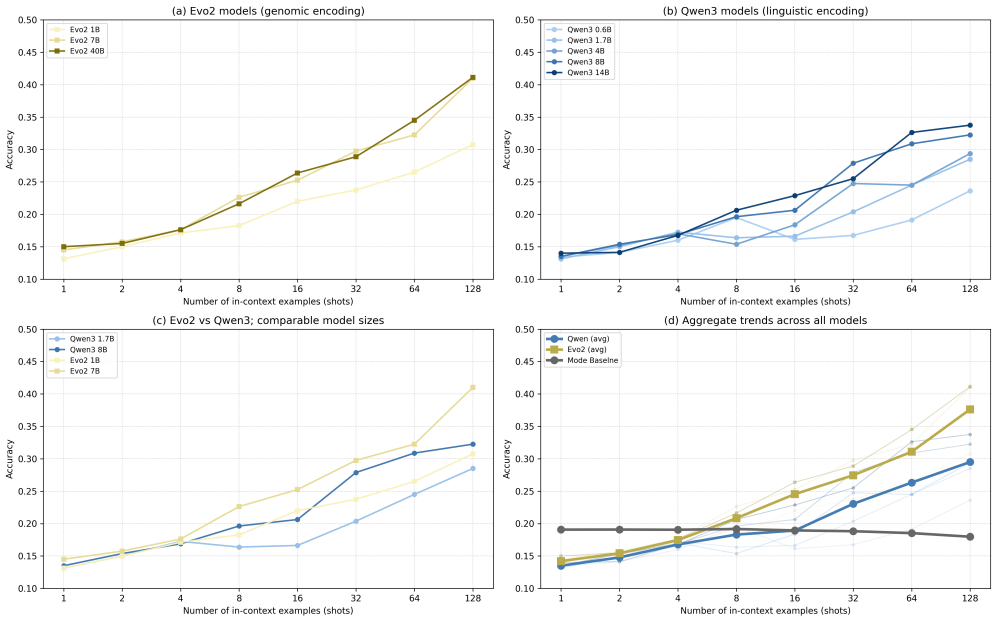


Figure 3: Few-shot performance of Qwen3 and Evo2 models. (a) Evo2 model performance with respect to log(shots). All models monotonically improve. 7B and 40B beat 1B. (b) Qwen3 model performance with respect to log(shots). All models improve, but not always monotonically. (c) At comparable sizes, Evo2 outperforms Qwen3. (d) Averaged performance across both model families. All models exceed the mode baseline shown in gray color. The exact accuracies and error bars (bootstrap-based standard errors) are included in §I.

.

Evo2 performs ICL on genomic tasks. In Fig. 2, we demonstrate that Evo2 7B can also perform few-shot in-context learning (ICL) on a relatively in-distribution binary genomic classification task: `human_nontata_promoters`, in which the model must predict whether a 251 nucleotide-long human DNA sequence is a promoter. Details of this experiment can be found in §K.

Further Analysis. We provide additional analyses of *BitLoad* (how many input bits influence the output) sensitivity in §D.1, *BitDiversity* (the number of minority bits in the output string) sensitivity in §L, and qualitative task differences in §D.2. Finally, we also provide analysis of how model scale impacts accuracy in §J.

4 Conclusion

We introduce a suite of bitstring reasoning tasks that can be encoded in both natural language and genomic sequences, showing that genomic models – like their linguistic counterparts – exhibit clear in-context learning. Across all Evo2 model sizes, we observe robust log-linear gains in accuracy with increasing demonstrations, paralleling the scaling trends of Qwen3 language models. These findings challenge the notion that ICL is unique to human language, suggesting it emerges whenever an expressive model is trained autoregressively on structured, pattern-rich data. See §A for a broader discussion of implications.

Potential future work: This work motivates searching for ICL in other non-linguistic modalities – time series (Das et al., 2024), system logs (Akhauri et al., 2025), physics simulations (Holzschuh et al., 2025), chess games (Ruoss et al., 2024), and climate projections (Duncan et al., 2025). Each offers a structured, patterned substrate that could support its own form of contextual reasoning. These diverse modalities, each with their unique structure and constraints, suggest a rich world of non-linguistic ICL capability waiting to be explored, and this work represents a maiden voyage into these extremely interesting waters.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems* (NeurIPS), 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/8ed3d610ea4b68e7afb30ea7d01422c6-Abstract-Conference.html.

Yash Akhauri, Bryan Lewandowski, Cheng-Hsi Lin, Adrian N. Reyes, Grant C. Forbes, Arissa Wongpanich, Bangding Yang, Mohamed S. Abdelfattah, Sagi Perel, and Xingyou Song. Performance prediction for large systems via text-to-text regression, 2025. URL https://arxiv.org/abs/2506.21718.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *International Conference on Learning Representations* (ICLR), 2022. URL https://arxiv.org/abs/2211.15661.

Arc Institute. Evo 2 1b base. https://huggingface.co/arcinstitute/evo2_1b_base, 2025a. Model card on Hugging Face.

Arc Institute. Evo 2 40b. https://huggingface.co/arcinstitute/evo2_40b, 2025b. Model card on Hugging Face.

Arc Institute. Evo 2 7b. https://huggingface.co/arcinstitute/evo2_7b, 2025c. Model card on Hugging Face.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. In *Annual Meeting of the Association for Computational Linguistics* (ACL), 2023. URL https://aclanthology.org/2023.acl-long.660/.

Gonzalo Benegas, Chengzhong Ye, Carlos Albors, Jianan Canal Li, and Yun S Song. Genomic language models: opportunities and challenges. *Trends in Genetics*, 2025.

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Mohsen Naghipourfar, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Gwanggyu Sun, Ali Taghibakshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Etowah Adams, Stephen A. Baccus, Steven Dillmann, Stefano Ermon, Daniel Guo, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R.K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, Jonathan C. Schmok, John St. John, Jeremy Sullivan, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modeling and design across all domains of life with evo 2. *bioRxiv*, 2025. doi: 10.1101/2025.02.18.638918. URL https://www.biorxiv.org/content/early/2025/02/21/2025.02.18.638918.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems* (NeurIPS), 2020a. URL https://arxiv.org/abs/2005.14165.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020b. URL https://arxiv.org/abs/2005.14165.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems* (NeurIPS), 2022. URL https://arxiv.org/abs/2205.05055.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. Parallel structures in pre-training data yield in-context learning. *ArXiv preprint*, abs/2402.12530, 2024. URL https://arxiv.org/abs/2402.12530.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/pdf/2110.14168.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

James P. C. Duncan, Elynn Wu, Surya Dheeshjith, Adam Subel, Troy Arcomano, Spencer K. Clark, Brian Henn, Anna Kwa, Jeremy McGibbon, W. Andre Perkins, William Gregory, Carlos Fernandez-Granda, Julius Busecke, Oliver Watt-Meyer, William J. Hurlin, Alistair Adcroft, Laure Zanna, and Christopher Bretherton. Samudrace: Fast and accurate coupled climate modeling with 3d ocean and atmosphere emulators, 2025. URL https://arxiv.org/abs/2509.12490.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. A complexity-based theory of compositionality. *ArXiv preprint*, abs/2410.14817, 2024a. URL https://arxiv.org/abs/2410.14817.

Eric Elmoznino, Tom Marty, Tejas Kasetty, Leo Gagnon, Sarthak Mittal, Mahan Fathi, Dhanya Sridhar, and Guillaume Lajoie. In-context learning and occam's razor. *ArXiv preprint*, abs/2410.14086, 2024b. URL https://arxiv.org/abs/2410.14086.

Zhouxiang Fang, Aayush Mishra, Muhan Gao, Anqi Liu, and Daniel Khashabi. ICL Ciphers: Quantifying "Learning" in In-Context Learning via Substitution Ciphers. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP), 2025. URL https://arxiv.org/abs/2504.19395.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics* (TACL), 2021. URL https://arxiv.org/abs/2101.02235.

Riccardo Grazzi, Julien Siems, Simon Schrodi, Thomas Brox, and Frank Hutter. Is mamba capable of in-context learning? *ArXiv preprint*, abs/2402.03170, 2024. URL https://arxiv.org/abs/2402.03170.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24:25, May 2023. doi: 10.1186/s12863-023-01123-8.

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

Damian Hodel and Jevin West. Response: Emergent analogical reasoning in large language models. *arXiv preprint arXiv:2308.16118*, 2023.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-transformer: Efficient and versatile transformers for physics simulations. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 23562–23602. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/holzschuh25a.html.

Zhongtao Jiang, Yuanzhe Zhang, Kun Luo, Xiaowei Yuan, Jun Zhao, and Kang Liu. On the in-context generation of language models. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP), 2024. doi: 10.18653/v1/2024.emnlp-main.568. URL https://aclanthology.org/2024.emnlp-main.568/.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Ivan Lee, Nan Jiang, and Taylor Berg-Kirkpatrick. Exploring the relationship between model architecture and in-context learning ability. *ArXiv preprint*, abs/2310.08049, 2023. URL https://arxiv.org/abs/2310.08049.

Martha Lewis and Melanie Mitchell. Evaluating the robustness of analogical reasoning in large language models. *arXiv preprint arXiv:2411.14215*, 2024.

Shuai Li, Zhao Song, Yu Xia, Tong Yu, and Tianyi Zhou. The closeness of in-context learning and weight shifting for softmax regression. *arXiv preprint arXiv:2304.13276*, 2023. URL https://arxiv.org/abs/2304.13276.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *International Conference on Machine Learning* (ICML), 2024. URL https://arxiv.org/pdf/2402.18819.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *ArXiv preprint*, abs/2307.03576, 2023. URL https://arxiv.org/abs/2307.03576.

Ziv Nevo and Ran El-Yaniv. On online learning of decision lists. *J. Mach. Learn. Res.*, 3:271–301, 2002. URL https://jmlr.org/papers/v3/nevo02a.html.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, July 2023. URL https://aclanthology.org/2023.findings-acl.527.

Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism. *ArXiv preprint*, abs/2306.04891, 2023. URL https://arxiv.org/abs/2306.04891.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *ArXiv preprint*, abs/2402.04248, 2024. URL https://arxiv.org/abs/2402.04248.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. URL https://openai.com/index/language-unsupervised/.

John Carlyle Raven, John H. Court, and John Carlyle Raven. Manual for raven's progressive matrices and vocabulary scales. 1962. URL https://api.semanticscholar.org/CorpusID:143337389.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *International Conference on Learning Representations* (ICLR), 2023. URL https://arxiv.org/pdf/2312.03002.

Jie Ren, Qipeng Guo, Hang Yan, Dongrui Liu, Quanshi Zhang, Xipeng Qiu, and Dahua Lin. Identifying semantic induction heads to understand in-context learning. *arXiv preprint arXiv:2402.13055*, 2024.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess, 2024. URL https://arxiv.org/abs/2402.04494.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy,

Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (TMLR), 2023. URL https://arxiv.org/abs/2206.04615.

Ramzan Kh. Umarov and Victor V. Solovyev. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PLOS ONE*, 12(2):e0171410, 2017. doi: 10.1371/journal.pone.0171410. URL https://doi.org/10.1371/journal.pone.0171410.

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 *Conference on Empirical Methods in Natural Language Processing* (EMNLP), 2023a. doi: 10.18653/v1/2023.emnlp-main.609. URL https://aclanthology.org/2023.emnlp-main.609.

Xiaolei Wang, Xinyu Tang, Wayne Xin Zhao, and Ji-Rong Wen. Investigating the pre-training dynamics of in-context learning: Task recognition vs. task learning. *ArXiv preprint*, abs/2406.14022, 2024. URL https://arxiv.org/abs/2406.14022.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. In *Advances in Neural Information Processing Systems* (NeurIPS), 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/3255a7554605a88800f4e120b3a929e1-Abstract-Conference.html.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541, 2023. URL https://arxiv.org/abs/2212.09196.

Taylor W Webb, Keith J Holyoak, and Hongjing Lu. Evidence from counterfactual tasks supports emergent analogical reasoning in large language models. *PNAS nexus*, 4(5):pgaf135, 2025.

Kevin Christian Wibisono and Yixin Wang. In-context learning from training on unstructured data: The role of co-occurrence, positional information, and training data structure. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL https://openreview.net/forum?id=Zvwwnfwxa4.

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* (ICLR), 2021. URL https://arxiv.org/abs/2111.02080.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

A Discussion

What are the implications of our findings on prior efforts to explain the emergence of ICL? First, let us organize the existing frameworks for pinpointing the conditions under which ICL emerges:

(E1) **ICL's emergence is due to data distributional properties:** The distributional properties of data, such as "parallel structures" in human language pretraining data (Chen et al., 2024), its compositional structure (Hahn & Goyal, 2023), "burstiness" (Chan et al., 2022) and other such properties (Wibisono & Wang, 2024; Reddy, 2023) may be of importance (and perhaps necessary) for the emergence of ICL.

(E2) **ICL's emergence is due to a compression mechanism:** The large-scale compression mechanism during massive pretraining might drive ICL (Elmoznino et al., 2024a;b; Hahn & Goyal, 2023).

(E3) **ICL's emergence may require specific architectural properties:** While Transformers might be better suited for ICL than LSTMs (Xie et al., 2021), evidence is mixed (Lee et al., 2023), and non-Transformer models have also demonstrated ICL capabilities (Grazzi et al., 2024; Park et al., 2024).

Our findings refine existing hypotheses about ICL's origins. The emergence of ICL in genomic models challenges accounts that rely solely on language-specific distributional structures (E1). The presence of ICL across both genomic and linguistic models supports the compression-based explanation (E2), suggesting that large-scale sequence compression and its induced inductive biases drive ICL across modalities. With respect to architecture (E3), Evo2 – an autoregressive hybrid combining convolutional and attention layers rather than a pure Transformer – exhibits similar scaling behavior, indicating that ICL does not depend on the pure Transformer form. Instead, architecture provides an expressive substrate that enables pattern induction once exposed to sufficiently large and structured data. Overall, these results position ICL as a modality-agnostic outcome of large-scale next-token prediction, rather than a phenomenon tied to linguistic statistics or a specific architecture.

What are the implications of our findings on the frameworks to explain how ICL operates? We next consider the major perspectives that seek to explain how ICL operates. One view holds that ICL functions as a mix of *task learning* and *task retrieval*, with demonstrations serving either to recall pretrained capabilities or to enable learning on the fly (Pan et al., 2023; Lin & Lee, 2024; Wang et al., 2024; Fang et al., 2025). Our symbolic reasoning tasks, instantiated in both linguistic and genomic domains, provide direct evidence for this *task learning* mode, aligning with this hypothesis and prior work (Pan et al., 2023; Fang et al., 2025). Because these tasks do not depend on pretrained semantic priors, they do *not* invoke task *retrieval*, offering limited insight into the Bayesian view that interprets ICL as implicit inference over latent concepts (Xie et al., 2021; Panwar et al., 2023; Wang et al., 2023b; Jiang et al., 2024). Meanwhile, our results remain agnostic toward the optimization-based hypothesis, which posits that ICL implements an implicit gradient-descent-like process (Akyürek et al., 2022; Ahn et al., 2023; Mahankali et al., 2023; Li et al., 2023), as well as the induction-based account, which attributes ICL to specialized "circuits" for performing inductive generalization (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2023a; Bansal et al., 2023; Ren et al., 2024). Together, our results most strongly support the presence of genuine task *learning* within ICL.

Does Evo2 have an innate advantage on these tasks? Possibly, for multiple reasons. First, though Evo2's is trained on less tokens total, all of Evo2's training tokens are long sequences of repeated nucleotides, and very little of Qwen3's training tokens are long sequences of repeated digits. Second, Evo2's StripedHyena2 architecture was found to to significantly outperform a vanilla transformer on long DNA sequences (Brixi et al., 2025). This could give Evo2 an innate advantage on long contexts containing the same few symbols vs Qwen3. Our result results do not imply Evo2's ICL ability is superior to Qwen3's, and we concede that our few-shot prompting setup may be biased toward Evo2. Attempts to make the task more legible to Qwen3, however, ran into the confound that Qwen3 has pretraining exposure to bitstring manipulation. Any prompting setup that included 0s and 1s immediately resulted in an extreme increase in Qwen3's performance. Future work is needed to establish a method that controls for Evo2's structural advantages without granting Qwen3 an unfair edge via its pretraining knowledge.

Why not test the models on semantic tasks? Semantic tasks – such as identifying the capitals associated with countries, classifying malformed proteins, etc. – require a fair amount of pretraining exposure to the concepts involved in the task as well as measuring ICL. While it's undoubtedly ICL when a model infers a semantic transformation (for instance, country→capital, word→opposite), the pretraining knowledge necessary to manifest this ICL precludes its use in extreme cross-modality comparisons. Focusing on far simpler bitstring transformations that can be learned entirely in-context allows for an apples-to-apples comparison between Evo2 and Qwen3. To ensure that the ICL observed is not an artifact of this simplified domain, we additionally demonstrate in Appendix K that Evo2 exhibits robust, scaling ICL on a native genomic task (promoter classification), though we exclude this from the main comparison to maintain parity with the linguistic models.

B Experimental Desiderata

Desideratum 1: Cross-domain comparability. Our experimental framework requires that each task be performable by *both* language and genomic models. Thus, every task must be representable in both linguistic and nucleotide alphabets. This constraint excludes existing benchmarks that rely on domain-specific semantics – such as language reasoning datasets (e.g., BIG-Bench, StrategyQA, GSM8K (Srivastava et al., 2023; Geva et al., 2021; Cobbe et al., 2021)) or biological tasks (e.g. variant effect prediction, exon identification (Brixi et al., 2025)). While one could, in principle, translate domain-specific tasks into an alternate alphabet (e.g., mapping language tokens to base sequences via quaternary encoding), doing so inherently biases the evaluation: the source domain retains advantages, while the target domain must operate on representations that are unnatural to it. As a result, such cross-domain encodings confound the comparison, reflecting representational translation artifacts rather than genuine differences in ICL behavior.

Desideratum 2: Limited vocabulary. Since genomic models operate on only four nucleotides (A, T, C, G), tasks must be expressible within an equally compact alphabet. This rules out the existing symbolic reasoning and analogy benchmarks (Hodel & West, 2023; Lewis & Mitchell, 2024; Webb et al., 2025), which rely on richer vocabularies (e.g., shapes, colors, or linguistic tokens with explicit semantic roles). Such tasks cannot be faithfully represented in a four-token regime without introducing artifacts or structural loss. Accordingly, our evaluation focuses on tasks that retain abstract reasoning structure while remaining compatible with the low-vocabulary symbolic space shared across linguistic and genomic models.

C Model Families and Selection Rationale

For fair cross-domain comparison, we use two representative model families: Qwen3 for human language and Evo2 for genomics. The rationale for this selection is as follows:

(a) *Parameter scaling:* Both families span multiple orders of magnitude in parameter count, enabling systematic scaling analysis of ICL ability. The Qwen3 series ranges from 0.6B to 14B parameters, while Evo2 includes 1B, 7B, and 40B models (Yang et al., 2025; Brixi et al., 2025). This parallel scaling structure facilitates consistent measurement of how ICL performance evolves with model size across linguistic and genomic modalities.

(b) *Compute matching:* The largest models in each family are trained with comparable total compute, offering an opportunity for an approximately compute-matched cross-domain comparison. Using the standard $6ND$ estimate (Kaplan et al., 2020), Qwen3-14B-Base is trained with about 3.2×10^{24} FLOPs, while Evo2-40B is trained with 2.25×10^{24} FLOPs (Yang et al., 2025; Brixi et al., 2025), making Evo2 uniquely well-suited for comparison with Qwen3 at scale.

(c) *Availability of base models:* The Qwen3 family releases base (*pre*-instruction-tuned) models at all scales up to 14B parameters, enabling direct evaluation of the intrinsic inductive reasoning ability of pure next-token predictors, without instruction-tuning artifacts. The Evo2 base models have no additional instruction tuning applied—the only nuance is that Evo2 1B is trained at a context length of 8192 nucleotides, whereas the 7B/40B are extended to a context length of one million.

(d) *Tokenizer:* Qwen3 uses a standard BPE tokenizer with a vocabulary size of 151,669. Evo2 uses a byte-level tokenizer, so individual nucleotides are mapped to individual tokens. Notably, Qwen's tokenizer maps single digits to single tokens, allowing for parity in our experiments (Yang et al., 2025; Brixi et al., 2025).

(e) *Context length:* Qwen3's 0.6B and 1.7B models have a context length of 32K. The remaining dense models have a context length of 128K. Evo2's 1B model has a context length of 8K, and the remaining models have a context length of 1M. As none of our experiments approach these limits, we can use all models in both families without fear of context length as a confound (Yang et al., 2025; Brixi et al., 2025).

(f) *Training corpora:* All Qwen3 models are trained on 36 trillion text tokens covering a wide variety of topics and languages. Evo1 1B is trained on 1 trillion tokens, Evo2 7B is trained on 2.4 trillion tokens, and Evo2 40B is trained on 9.3 trillion tokens. These extensive training corpora ensure that any potential ICL dynamics have thoroughly emerged (Yang et al., 2025; Brixi et al., 2025).

(g) *Architecture:* Qwen3 is based on a conventional Llama-like transformer architecture, while Evo2 uses the StripedHyena2 architecture which intersperses convolutional layers with attention-based ones. While ideally both model families would use the same architecture, there were no vanilla transformers at Evo2's compute scale (Yang et al., 2025; Brixi et al., 2025).

(h) *Licensing:* All models are released under Apache 2.0 (Yang et al., 2025; Arc Institute, 2025a;c;b), ensuring reproducibility.

D Supplementary Analyses of Task Complexity and Qualitative Differences

D.1 ICL Sensitivity to Task Complexity: BitLoad Analysis

To understand which transformations Qwen3 and Evo2 infer most effectively, we analyze their performance across varying task complexities. We focus on the largest models in each family (Qwen3-14B and Evo2-40B) under the ($n = 128$) shot regime, providing both models ample opportunity to display their ICL abilities.

Defining BitLoad. We introduce *BitLoad*, a measure of a function's intrinsic complexity. Informally, BitLoad quantifies how many input bits influence the output. Formally, it is defined as:

$$\text{BitLoad}(f) = \sum_{i=1}^{k} \mathbb{1}\Big[\exists x, j : f_j(x) \neq f_j(x^{\oplus i})\Big], \tag{2}$$

where $x^{\oplus i}$ denotes x with bit i flipped, and $f_j(x)$ returns the j-th output bit. Intuitively, it counts the number of bit positions whose perturbation changes the output. So, the larger the BitLoad of a function, the harder it is since it requires the model to attend to more bits. This metric is similar to the existing statistical measure of "relevant features", just defined specifically for our bitstring manipulation tasks. (Nevo & El-Yaniv, 2002).

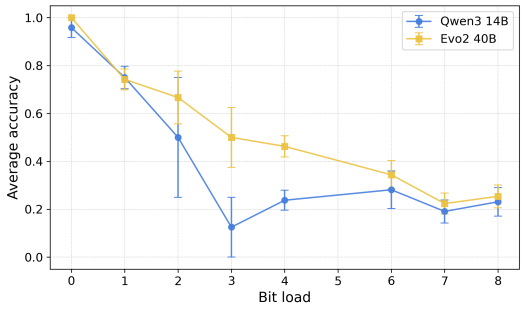


Figure 4: Accuracy vs. BitLoad averaged across all tasks (BitLoad; Eq.2). Qwen declines sharply with increasing BitLoad, while Evo degrades more gradually, indicating greater robustness. Details in §D.1.

Fig. 4 shows the mean accuracy of Qwen vs Evo with respect to the BitLoad of all of our tasks. A full table of the BitLoad of every tested task is attached in the appendix E.3. We see that both models achieve near-perfect accuracy on constant (0 BitLoad) tasks and remain similar at BitLoad 1. However, by BitLoad 2, Evo begins to outperform Qwen, and beyond that point Qwen's accuracy drops sharply – falling below 20% by BitLoad 4 – while Evo's performance declines more gradually, remaining above 40% before converging near 20% at BitLoad 8. This asymmetry suggests that Evo

maintains partial generalization as dependency depth increases (BitLoad value), whereas Qwen's ICL collapses more abruptly.

While BitLoad strongly correlates with task accuracy, Fig. 4 also shows notable deviations within the error bars. Thus, while task complexity is a strong predictor of ICL accuracy, other factors (e.g., transformation depth, pretraining exposure) likely play meaningful roles, which motivate our qualitative study in §D.2.

D.2 Qualitative Analysis of Functional and Behavioral Differences

To complement the quantitative BitLoad analysis (§D.1), we conduct a per-task comparison to identify qualitative differences between Qwen3 and Evo2. Specifically, we analyze which tasks each model performs best on and how their inductive profiles diverge. We focus on the $n = 128$ shot regime for the largest models – Qwen3-14B and Evo2-40B – where both have maximal opportunity to exhibit ICL. We rank all tasks by model accuracy and examine the top ten for each model. Tasks that appear in one model's top ten but not the other are considered "exclusive."

Exclusive competencies. Qwen's exclusive tasks involve right-shift operations: `"spread_last_bit"` $\rightarrow$ `"shift_right_zero"` and `"edge_mask"` $\rightarrow$ `"shift_right_zero"`. For instance, "`shift_right_zero`" pads the bitstring on the left with a zero and truncates the last bit (e.g., $01010000 \rightarrow 00101000$). Qwen achieves 100% accuracy on `"spread_last_bit"` $\rightarrow$ `"shift_right_zero"`, whereas Evo2 achieves only 50%. A similar but smaller gap appears for `"edge_mask"` $\rightarrow$ `"shift_right_zero"` (87.5% vs. 62.5%).

In contrast, Evo2's exclusive strengths involve multi-bit transformations. A clear example is `"flip_bits"` $\rightarrow$ `"right_half"`, which applies bitwise NOT followed by masking the first half of the input (e.g., $01011100 \rightarrow 00000011$). Evo2 achieves 87.5% accuracy, while Qwen only 25%. This task has a BitLoad of four, consistent with Evo's superior performance on medium-complexity (2–4 bit) transformations observed in Fig. 4.

Shared strengths. Despite these differences, 7 of the top 10 tasks are shared between Qwen and Evo, yielding an intersection-over-union of 0.54. Both models excel at simple transformations such as constant-outputs or single-bit dependencies, e.g., `"spread_last_bit"` which copies the final bit to all positions.

Differential skill profiles. To sharpen contrasts, we identify the ten tasks most favoring each model. Qwen's advantages are concentrated in simple shifts and aggregation tasks. It outperforms Evo by 37.5% on the `"minority"` operation (output all 1s if zeros $>$ ones), and by a similar margin on two parity-based tasks requiring counting the number of 1s. These trends suggest that Qwen may be better at reasoning over global properties of bitstrings. Its superiority on simple shifts may also be explained by the extreme rarity of frame shift mutations in DNA due to how catastrophic they are – a single nucleotide offset can decimate an entire protein. This is empirically supported by the fact that single-nucleotide deletions/shifts increase perplexity far more than other common mutations when presented to Evo2 (Brixi et al., 2025).

Evo2, by contrast, dominates tasks requiring full-bitstring manipulation. It achieves 62.5% on bitwise NOT (vs. 0% for Qwen), 62.5% on identity (vs. 12.5%), and large margins on compositions such as `"flip_bits"` $\rightarrow$ `"right_half"` (87.5% vs. 25%) and `"rotl1"` $\rightarrow$ `"flip_bits"` (37.5% vs. 0%). This exposes perhaps the most important difference between Qwen and Evo's ICL in this specific context: Evo can learn simple full-bitstring operations in-context, whereas Qwen cannot. Notably, Qwen3's base models are trivially capable of learning the identity in a more familiar few-shot context – when examples are presented with arrows and newlines separating them, instead of our intentionally unfamiliar encoding. Thus these results should be taken as an existence proof of ICL in Evo2, not a definitive statement of Evo2 having more ICL ability than Qwen. We leave a comparison of these models across broader tasks to future work.

E Further Details on Synthetic Task Definition

E.1 Table of Primitives

The following table describes the thirty primitives used to construct the task space via composition – their use is described in §2.1.

Primitive	Description
`alternating_start_one`	Produce a mask marking positions that differ from the alternating 1010... pattern starting with 1 (1 = mismatch, 0 = match).
`alternating_start_zero`	Produce a mask marking positions that differ from the alternating 0101... pattern starting with 0 (1 = mismatch, 0 = match).
`center_mask`	Zero out the first and last bits while leaving the interior bits unchanged; strings of length ≤ 2 become all zeros.
`double_rotl`	Circularly rotate the bitstring two positions to the left.
`double_rotr`	Circularly rotate the bitstring two positions to the right.
`edge_mask`	Preserve the first and last bits and zero out every interior bit (length 0/1 strings pass through).
`flip_bits`	Invert every bit, swapping 0s for 1s and vice versa.
`identity`	Return the bitstring unchanged.
`invert_prefix`	Flip the bits in the left half of the string, keep the right half as is.
`invert_suffix`	Keep the left half as is and flip every bit in the right half of the string.
`keep_even_positions`	Keep bits at even indices (0-based) and zero out bits at odd indices.
`keep_odd_positions`	Keep bits at odd indices (0-based) and zero out bits at even indices.
`left_half`	Preserve the left half of the string and replace the right half with zeros.
`majority`	Fill the string with the majority bit from the input; ties resolve to all 1s.
`meta_constant`	Returns a random, pre-set constant.
`minority`	Fill the string with the minority bit from the input; ties resolve to all 0s.
`mirror_half`	Copy the left half of the string onto the right half in reverse order, keeping the center bit unchanged for odd lengths.
`ones_if_palindrome`	Output all 1s if the input is a palindrome; otherwise output all 0s.
`parity_fill`	Output all 1s when the input contains an odd number of 1s; otherwise output all 0s.
`reverse_bits`	Reverse the order of the bits in the string.
`right_half`	Zero out the left half and keep the right half unchanged.
`rotl1`	Circularly rotate the bitstring one position to the left.
`rotr1`	Circularly rotate the bitstring one position to the right.
`shift_left_zero`	Shift the string left by one, dropping the first bit and appending a 0 on the right.
`shift_right_zero`	Shift the string right by one, inserting a 0 on the left and dropping the last bit.
`spread_first_bit`	Replace every position with the first bit of the input.
`spread_last_bit`	Replace every position with the last bit of the input.
`swap_halves`	Swap the left and right halves of the string.
`swap_pairs`	Swap each adjacent pair of bits (positions 0/1, 2/3, ...).
`xor_with_s0`	Can only be applied after another primitive. Computes the logical XOR between the original input s_0 and the output of the first primitive.

Table 1: The 30 unary primitives used to construct functions in F.

E.2 TABLE OF FUNCTIONS

The following table describes the one hundred specific compositions of primitives used to construct the evaluation suite in §**??**.

Function	Function	Function
identity	spread_last_bit	swap_halves → shift_left_zero
rotl1	invert_prefix	swap_halves → shift_right_zero
reverse_bits	invert_suffix	shift_left_zero → swap_halves
flip_bits	meta_constant	shift_right_zero → swap_halves
swap_halves	flip_bits → reverse_bits	keep_even_positions → flip_bits
majority	rotl1 → reverse_bits	keep_odd_positions → flip_bits
minority	reverse_bits → rotl1	flip_bits → keep_even_positions
parity_fill	rotl1 → flip_bits	flip_bits → keep_odd_positions
alternating_start_one	swap_halves → reverse_bits	edge_mask → flip_bits
alternating_start_zero	swap_halves → flip_bits	center_mask → flip_bits
left_half	double_rotl → flip_bits	shift_left_zero → keep_even_positions
right_half	rotr1 → flip_bits	shift_left_zero → keep_odd_positions
double_rotl	spread_first_bit → flip_bits	shift_right_zero → keep_even_positions
rotr1	spread_last_bit → flip_bits	shift_right_zero → keep_odd_positions
double_rotr	left_half → flip_bits	keep_even_positions → reverse_bits
ones_if_palindrome	right_half → flip_bits	keep_odd_positions → reverse_bits
mirror_half	flip_bits → left_half	shift_left_zero → parity_fill
spread_first_bit	flip_bits → right_half	shift_right_zero → parity_fill
spread_last_bit	double_rotl → reverse_bits	parity_fill → shift_left_zero
invert_prefix	rotl1 → swap_halves	parity_fill → shift_right_zero
invert_suffix	xor_with_s0	spread_first_bit → shift_left_zero
meta_constant	flip_bits → xor_with_s0	spread_last_bit → shift_right_zero
shift_left_zero	ones_if_palindrome → flip_bits	spread_first_bit → keep_even_positions
shift_right_zero	flip_bits → mirror_half	spread_last_bit → keep_odd_positions
swap_pairs	invert_prefix → reverse_bits	spread_first_bit → edge_mask
keep_even_positions	left_half → reverse_bits	spread_last_bit → edge_mask
keep_odd_positions	right_half → reverse_bits	spread_first_bit → center_mask
edge_mask	parity_fill → flip_bits	spread_last_bit → center_mask
center_mask	rotl1 → spread_first_bit	rotl1 → shift_left_zero
xor_with_s0	shift_left_zero → flip_bits	rotl1 → shift_right_zero
flip_bits → reverse_bits	shift_right_zero → flip_bits	shift_left_zero → rotl1
rotl1 → reverse_bits	flip_bits → shift_left_zero	shift_right_zero → rotl1
reverse_bits → rotl1	flip_bits → shift_right_zero	reverse_bits → edge_mask
rotl1 → flip_bits	swap_pairs → flip_bits	reverse_bits → center_mask
swap_halves → reverse_bits	shift_left_zero → reverse_bits	edge_mask → shift_left_zero
swap_halves → flip_bits	shift_right_zero → reverse_bits	edge_mask → shift_right_zero
double_rotl → flip_bits	spread_first_bit → flip_bits	shift_left_zero → shift_left_zero
rotr1 → flip_bits	shift_left_zero → edge_mask	shift_left_zero → swap_pairs

Table 2: The complete set of 100 functions in F, consisting of 30 single primitives and 70 composed functions ($(f \to g)(x) = g(f(x))$).

E.3 BitLoad of Functions

Function / Composition	BitLoad	Function / Composition	BitLoad
identity	8	rotl1	8
reverse_bits	8	flip_bits	8
swap_halves	8	majority	8
minority	8	parity_fill	8
alternating_start_one	8	alternating_start_zero	8
left_half	4	right_half	4
double_rotl	8	rotr1	8
double_rotr	8	ones_if_palindrome	8
mirror_half	4	spread_first_bit	1
spread_last_bit	1	invert_prefix	8
invert_suffix	8	meta_constant	0
flip_bits → reverse_bits	8	rotl1 → reverse_bits	8
reverse_bits → rotl1	8	rotl1 → flip_bits	8
swap_halves → reverse_bits	8	swap_halves → flip_bits	8
double_rotl → flip_bits	8	rotr1 → flip_bits	8
spread_first_bit → flip_bits	1	spread_last_bit → flip_bits	1
left_half → flip_bits	4	right_half → flip_bits	4
flip_bits → left_half	4	flip_bits → right_half	4
double_rotl → reverse_bits	8	rotl1 → swap_halves	8
xor_with_s0	0	flip_bits → xor_with_s0	0
ones_if_palindrome → flip_bits	8	flip_bits → mirror_half	4
invert_prefix → reverse_bits	8	left_half → reverse_bits	4
right_half → reverse_bits	4	parity_fill → flip_bits	8
rotl1 → spread_first_bit	1	shift_left_zero	7
shift_right_zero	7	swap_pairs	8
keep_even_positions	4	keep_odd_positions	4
edge_mask	2	center_mask	6
shift_left_zero → flip_bits	7	shift_right_zero → flip_bits	7
flip_bits → shift_left_zero	7	flip_bits → shift_right_zero	7
swap_pairs → flip_bits	8	shift_left_zero → reverse_bits	7
shift_right_zero → reverse_bits	7	swap_halves → shift_left_zero	7
swap_halves → shift_right_zero	7	shift_left_zero → swap_halves	7
shift_right_zero → swap_halves	7	keep_even_positions → flip_bits	4
keep_odd_positions → flip_bits	4	flip_bits → keep_even_positions	4
flip_bits → keep_odd_positions	4	edge_mask → flip_bits	2
center_mask → flip_bits	6	shift_left_zero → keep_even_positions	4
shift_left_zero → keep_odd_positions	3	shift_right_zero → keep_even_positions	3
shift_right_zero → keep_odd_positions	4	keep_even_positions → reverse_bits	4
keep_odd_positions → reverse_bits	4	shift_left_zero → parity_fill	7
shift_right_zero → parity_fill	7	parity_fill → shift_left_zero	8
parity_fill → shift_right_zero	8	spread_first_bit → shift_left_zero	1
spread_last_bit → shift_right_zero	1	spread_first_bit → keep_even_positions	1
spread_last_bit → keep_odd_positions	1	spread_first_bit → edge_mask	1
spread_last_bit → edge_mask	1	spread_first_bit → center_mask	1
spread_last_bit → center_mask	1	rotl1 → shift_left_zero	7
rotl1 → shift_right_zero	7	shift_left_zero → rotl1	7
shift_right_zero → rotl1	7	reverse_bits → edge_mask	2
reverse_bits → center_mask	6	edge_mask → shift_left_zero	1
edge_mask → shift_right_zero	1	shift_left_zero → shift_left_zero	6
shift_left_zero → swap_pairs	7	shift_left_zero → edge_mask	1

Table 3: BitLoad for every primitive and composed function in F. Used in §D.1.

F A Global Metric Over All Programs

For a given function f, model M, and number of in-context examples n, we define the average accuracy over all context sets $E_N = \{E \subset S : |E| = n\}$:

$$A_f(M, n) = \frac{1}{|E_N|(|S| - n)} \sum_{E \in E_N} \sum_{x \in S \setminus E} A(f, E, x, M).$$

The overall benchmark score across all functions F is then

$$P(M, n) = \frac{1}{|F|} \sum_{f \in F} A_f(M, n).$$

Because exact evaluation is intractable (for $n = 8$, $|E_N|(|S| - N) = \binom{|S|}{n}(|S| - n) = \binom{256}{8} \cdot 248 \approx 1.016 \times 10^{17}$) we estimate $A_f(M, n)$ via Monte Carlo, as discussed in §2.2.

G Prompt Encoding

Prompts are encoded using a unified symbolic scheme to enable evaluation across linguistic and genomic models. For linguistic models, bits $(0, 1)$ are mapped to two random digits (0–9); for genomic models, to random nucleotides $(\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G})$. The separator token $(\rightarrow)$ is omitted, and in-context examples are separated by a randomly chosen unused token distinct from those representing 0 and 1. (See Fig. 1 for examples.) All mappings are randomized per trial to avoid memorization or positional bias.

H Mode Baseline

We define a mode baseline that always predicts the most frequent output observed in the context. For a given function f, context set $E \subset S$, and query input $x \in S \setminus E$, the mode prediction is $\hat{y}_{\text{mode}}(f, E, x) = \arg\max_{y \in S} \big|\{\, e \in E : f(e) = y \,\}\big|$, where ties in the $\arg\max$ are broken randomly. This simply corresponds to guessing the most common output in the few-shot examples. The overall mode baseline with n shots and across the set of all functions F is:

$$\hat{P}_{\text{mode}}(n) = \frac{1}{m|F|} \sum_{f \in F} \sum_{t=1}^{m} \mathbb{1}\big[\hat{y}_{\text{mode}}(f, E^{(t)}, x^{(t)}) = f(x^{(t)})\big], \tag{3}$$

where $E^{(t)}$ and $x^{(t)}$ are sampled identically to the model evaluation in Eq.1. This baseline corresponds to making an educated guess based only on the overall distribution of function outputs that simply learns the majority statistics of the prompt without attempting to infer the underlying transformation.

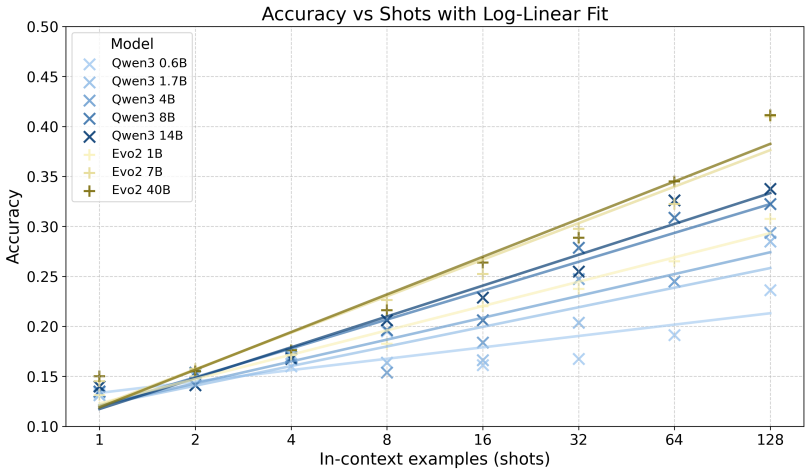
I Full Accuracy Results

Here we show the full table of accuracies used in §3.1 to analyze the ICL capabilities of Evo2 and Qwen3 and perform the necessary statistical tests.

Model	1 Shot	2 Shots	4 Shots	8 Shots	16 Shots	32 Shots	64 Shots	128 Shots
Qwen3 0.6B	$13.4_{\pm 2.3}$	$14.1_{\pm 2.3}$	$16.0_{\pm 2.6}$	$19.5_{\pm 2.7}$	$16.1_{\pm 2.4}$	$16.8_{\pm 2.5}$	$19.1_{\pm 2.7}$	$23.6_{\pm 2.9}$
Qwen3 1.7B	$13.1_{\pm 2.2}$	$15.0_{\pm 2.7}$	$\mathbf{17.2}_{\pm 2.8}$	$16.4_{\pm 2.6}$	$16.6_{\pm 2.7}$	$20.4_{\pm 2.9}$	$24.5_{\pm 3.4}$	$28.5_{\pm 3.5}$
Qwen3 4B	$13.5_{\pm 2.4}$	$15.2_{\pm 2.4}$	$17.0_{\pm 2.6}$	$15.4_{\pm 2.5}$	$18.4_{\pm 2.7}$	$24.8_{\pm 3.2}$	$24.5_{\pm 3.4}$	$29.4_{\pm 3.4}$
Qwen3 8B	$13.5_{\pm 2.3}$	$\mathbf{15.4}_{\pm 2.5}$	$16.9_{\pm 2.7}$	$19.6_{\pm 2.7}$	$20.6_{\pm 2.8}$	$\mathbf{27.9}_{\pm 3.3}$	$30.9_{\pm 3.5}$	$32.2_{\pm 3.5}$
Qwen3 14B	$\mathbf{14.0}_{\pm 2.4}$	$14.1_{\pm 2.4}$	$16.8_{\pm 2.7}$	$\mathbf{20.6}_{\pm 2.9}$	$\mathbf{22.9}_{\pm 3.1}$	$25.5_{\pm 3.2}$	$\mathbf{32.6}_{\pm 3.7}$	$\mathbf{33.8}_{\pm 3.5}$
Evo2 1B	$13.1_{\pm 2.2}$	$15.0_{\pm 2.4}$	$17.1_{\pm 2.9}$	$18.2_{\pm 2.7}$	$22.0_{\pm 2.9}$	$23.8_{\pm 2.8}$	$26.5_{\pm 3.0}$	$30.8_{\pm 3.1}$
Evo2 7B	$14.5_{\pm 2.4}$	$\mathbf{15.8}_{\pm 2.7}$	$\mathbf{17.6}_{\pm 2.9}$	$\mathbf{22.6}_{\pm 2.9}$	$25.2_{\pm 3.0}$	$\mathbf{29.8}_{\pm 3.1}$	$32.2_{\pm 3.1}$	$41.0_{\pm 3.4}$
Evo2 40B	$\mathbf{15.0}_{\pm 2.6}$	$15.5_{\pm 2.5}$	$\mathbf{17.6}_{\pm 2.8}$	$21.6_{\pm 3.1}$	$\mathbf{26.4}_{\pm 3.1}$	$28.9_{\pm 3.1}$	$\mathbf{34.5}_{\pm 3.2}$	$\mathbf{41.1}_{\pm 3.3}$

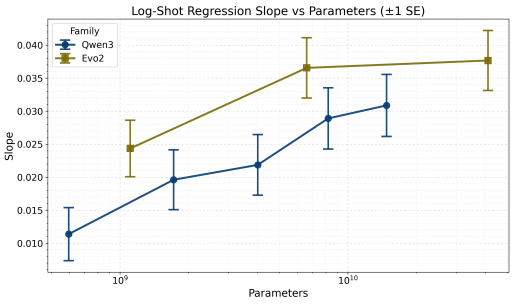
Table 4: In-context learning performance across model families and shot counts. Values show accuracy ± standard error. All numbers are percentages. **Bold** numbers show the best performance within a model family. Models are ordered by parameter count within each family.

J META-REGRESSION FOR ICL EFFICACY WITH NUMBER OF DEMONSTRATIONS



(a) Few-shot performance of Qwen3 and Evo2 models. All models show consistent linear improvement with respect to $\log(\text{shots})$. In contrast, no such improvement occurs for the naive baseline.

(b) Baseline accuracy decreases slightly with scale as the model gains more parameters for both Evo2 and Qwen3.



(c) ICL rate vs. model size: sharp gains up to 4B for Qwen3; mild boost from Evo2 1B to 7B; both plateau after 4–7B.

Figure 5: Few-shot behavior and scaling trends across Qwen3 and Evo2.

We perform linear regressions to predict accuracy from shot count with each model. For each model M, fit the linear regression: $\hat{P}(M, n) = \alpha_0(M) + \alpha_1(M)\log(n) + \varepsilon$. The raw regressions are shown in Fig. 5a. Predictably, all α_1 are positive as all models are capable of learning in-context.

We can interpret α_0 as representing the model's base accuracy at the task, what it would logically achieve with only one shot to identify the task. We can then interpret α_1 as the model's ICL efficacy: the speed at which it adapts to the task being presented and at which its accuracy improves. Analyzing how these values change across parameter values reveals insights into the ICL abilities of both the Qwen3 and Evo2 models.

First, we analyze how α_0 changes in each model family – this analysis can be seen in Fig. 5b. Both Evo2's and Qwen3's initial α_0 remains essentially constant model-to-model, indicating that all models have similar levels of few-shot baseline performance. Notably, Evo2 and Qwen3 have essentially identical intercepts at around 0.12. This implies that despite drastically different training data, the overall amount of prior knowledge the models have coming into this task is roughly similar. This rules out Qwen simply having 'less experience' with this sort of task.

If one looks at α_1 – ICL efficacy – in Fig. 5c, a dramatically different picture is painted. Here, both Qwen3 and Evo2 follow similar patterns with a significant difference. Qwen3's ICL efficacy increases monotonically with respect to parameters, more than doubling from the 0.6B to the 14B. Evo2 follows suit (albeit less dramatically), with ICL efficacy monotonically increasing from the 1B to the 40B.

In absolute terms, however, Evo2, when parameter-matched, adapts in-context faster than Qwen3 does. Evo2 40B outperforms Qwen3 14B significantly, and it takes until Qwen3 8B to exceed the ICL ability of Evo2 1B.

Taken together, this data suggests that Qwen3 and Evo2 have similar amounts of pretraining exposure to be able to solve these tasks, and that Evo2 simply has better overall ICL capability (in this regime) – even though Qwen's ICL ability increases more rapidly with respect to parameters.

K Demonstration of Evo2's ICL Ability on a Genomic Task

We demonstrate that Evo2 7B can perform few-shot in-context learning (ICL) on `human_nontata_promoters`, a binary genomic classification task in which the model must predict whether a 251 nucleotide-long human DNA sequence is a promoter. To make the task non-trivial, sequences in the positive class are restricted to non-TATA promoters, removing the "TATA box", a pattern that serves as an obvious, exploitable marker (Umarov & Solovyev, 2017; Grešová et al., 2023).

For each test query, we construct a few-shot prompt by concatenating balanced examples from the training set (equal numbers of promoters and non-promoters), followed by a held-out test sequence. Each training example is immediately followed by an encoded label. The label consists of a 24 nucleotide buffer (a single nucleotide repeated 24 times) and then a 24 nucleotide label run (another nucleotide repeated 24 times). The buffer nucleotide and the two label nucleotides (for class 0 vs. class 1) are re-sampled uniformly at random without replacement from $\{A, C, G, T\}$ for every trial, so the mapping changes across evaluations. This prevents any nucleotide-specific biasing effects.

To predict the test label, we score two candidate prompts - one where the test sequence is followed by the class-0 label and one followed by the class-1 label - and choose the label with lower perplexity on the label run (i.e., computed only over the final 24 label nucleotides, excluding the buffer). This lets us measure whether Evo2 can infer the label mapping from the few-shot context and apply it to the query sequence.

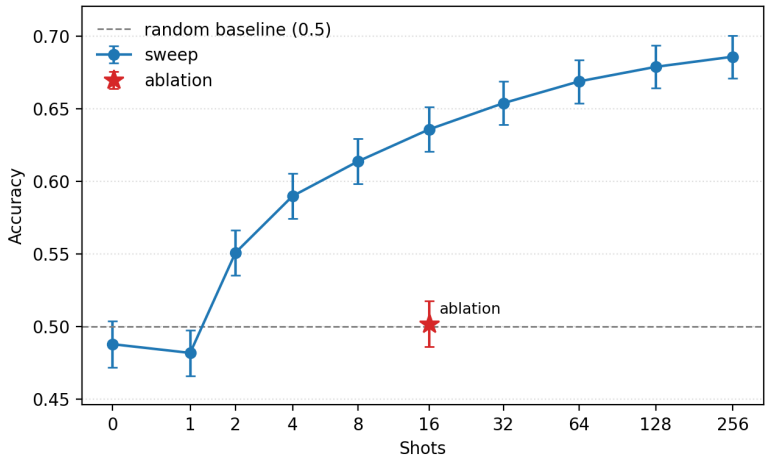


Figure 6: Performance of Evo2 7B on human_nontata_promoters at varying shot counts. Error bars are ± one standard error. Evo2 7B's performance shows clear monotonic improvements with respect to shot count.

In Figure 6, we observe monotonic improvement with respect to shot count, with accuracy rising from below random baseline at 1-shot (48.2%) to 63.6% at 16-shot. These gains further rise to 68.6% after 256 shots. A saturated power law achieves an R^2 of 0.998 in the 2-256 shot regime, which is characteristic of few-shot ICL. Furthermore, we show via ablation that the improvement is due to correctly-labeled examples presented in context—randomly permuting the labels in the 16-shot regime drops performance from 63.6% back to random chance (50.2%).

L Analysis of ICL with Program Synthesis Tasks: Bit-Diversity

BitLoad captures a measure of theoretical information bottleneck required to make predictions in program synthesis tasks. However, this does not always correlate with what models find easy to do. In Fig. 8 and Fig. 9, we plot the accuracy of Evo and Qwen models with respect to BitLoad. We notice that few high BitLoad tasks have high accuracy compared to all medium BitLoad tasks, illustrating that some high BitLoad tasks can be easy for these models to solve, probably due to the nature of patterns found during pre-training.

To further analyze the nature of ICL exhibited through these tasks, we define BitDiversity as *the number of minority bits in the output string*. In Fig. 10 and Fig. 11, we plot the accuracy with respect to BitDiversity. These plots try to estimate the effect of entropy in the output on model performance, and we see a more expected trend: models tend to perform better on low-entropy outputs, regardless of BitLoad. However, bin-wise performance trends are always increasing with shots, supporting the central hypothesis that ICL increases with increasing number of demos in these models.

Prevalence of 0-BitDiversity outputs. Looking at the expected and predicted outputs of trials from our tasks (Fig. 12), we made a few interesting observations about 0-BitDiversity (BD) outputs.

- Around 25% of true targets are 0-BD. This is a significantly high number as we only have 2 0-BD bit strings in all possible bit strings of any length K. It implies that many of our tasks create low BitDiversity outputs on random inputs, i.e., 0-BD outputs.
- Models tend to produce a lot of 0-BD outputs, much higher than the number of 0-BD true targets in the low-shot regime. But this number quickly drops to the expected number with higher shots.
- A large portion of the baseline (1-shot) performance can be explained by this prevalence of 0-BD outputs, but with more shots we get stronger evidence of ICL with increasing correctly predicted non 0-BD cases.

Understandable Mistakes. A potential confound is what we will call "understandable mistakes". These mistakes occur when the model outputs an incorrect answer that would be correct for some other programs given the few-shot context. Formally, a model's output y (see 2.2 for notation):

$$y = M\Big(e_1 \to f(e_1), e_2 \to f(e_2), \ldots, e_N \to f(e_N), x\Big)$$

is an **understandable mistake** if $y \neq f(x)$ but there exists $f_2 \in F$ such that $\forall e_i \quad f(e_i) = f_2(e_i)$ and $y = f_2(x)$.

These understandable mistakes are an alarming confound at low shot counts, but their effect vanishes by $N = 16$. They occur most in tasks with low BitDiversity, which can often be confused with each other. Figure 7 below shows how understandable mistakes in Qwen3-4B's inferences decay exponentially as the number of shots increases.

Noise due to Monte Carlo Trials. We use $m = 8$ for our per-function Monte Carlo trials when we compute accuracy. This has little aggregate impact when comparing model performance across the entire suite, but drastically reduces the significance of results when comparing models and trends at the task-level. We only have eight discrete bands of accuracy at which we can estimate a model's per-task performance – which reduces the expressiveness of regression. Worse, this can lead to models getting a lucky prompt or two with a 0-BD output, which raises performance to 12.5% or 25% without the model truly understanding the task. Addressing these confounds would simply require increasing m by an order of magnitude or so. Alternatively, tasks of interest could be identified and m selectively increased for those tasks to enable more nuanced analysis.

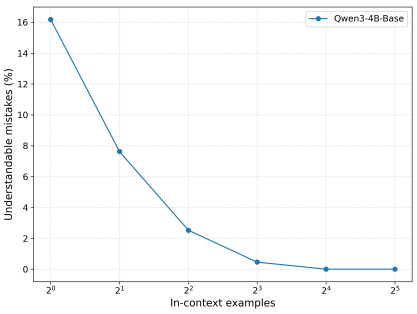


Figure 7: Qwen3-4B's rate of understandable mistakes with respect to the number of shots. Despite starting at 16% in the one-shot regime, they fall to less than 1% by 8 shots and vanish entirely at 32 shots. This underscores how understandable mistakes are only a confound at very low shot counts and can essentially be ignored past 4 shots.

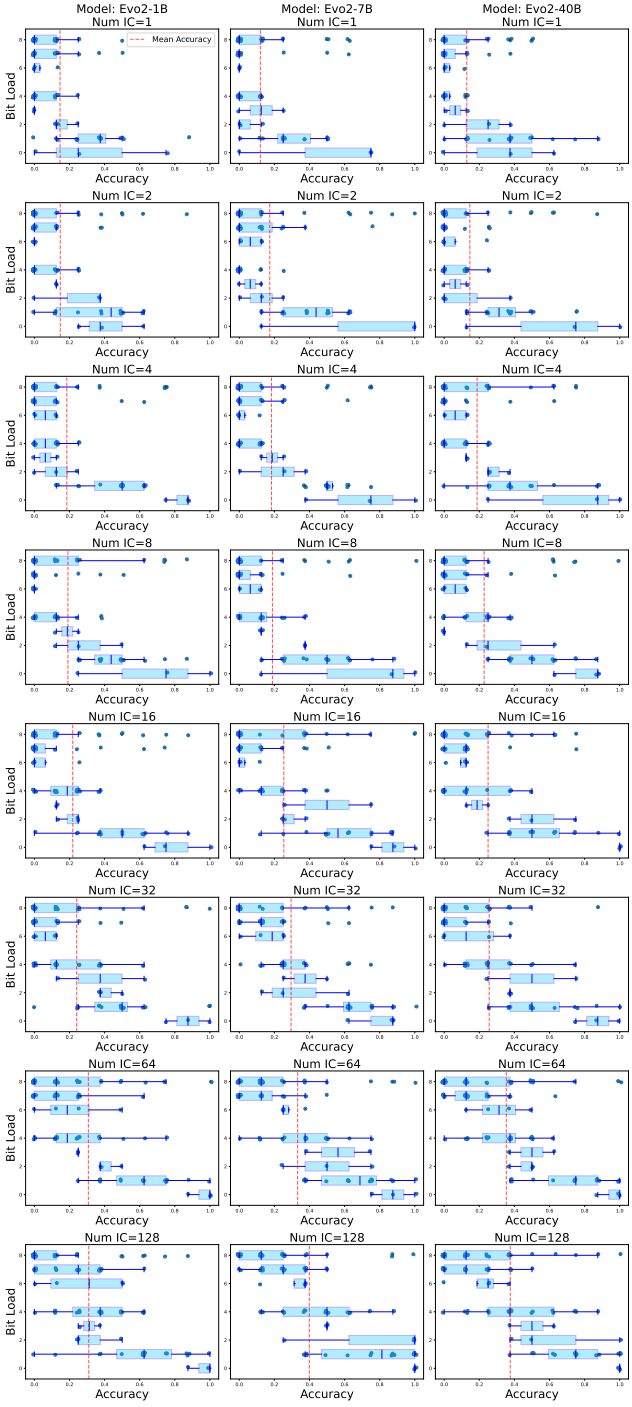


Figure 8: BitLoad vs Accuracy for all Evo models at all ICL shot-numbers. The per-BitLoad distribution of model performance at each shot level shows an uneven affinity of models for solving tasks at different BitLoads. However, all trends support our overall hypotheses.

Figure 9: BitLoad vs Accuracy for all Qwen models at all ICL shot-numbers.

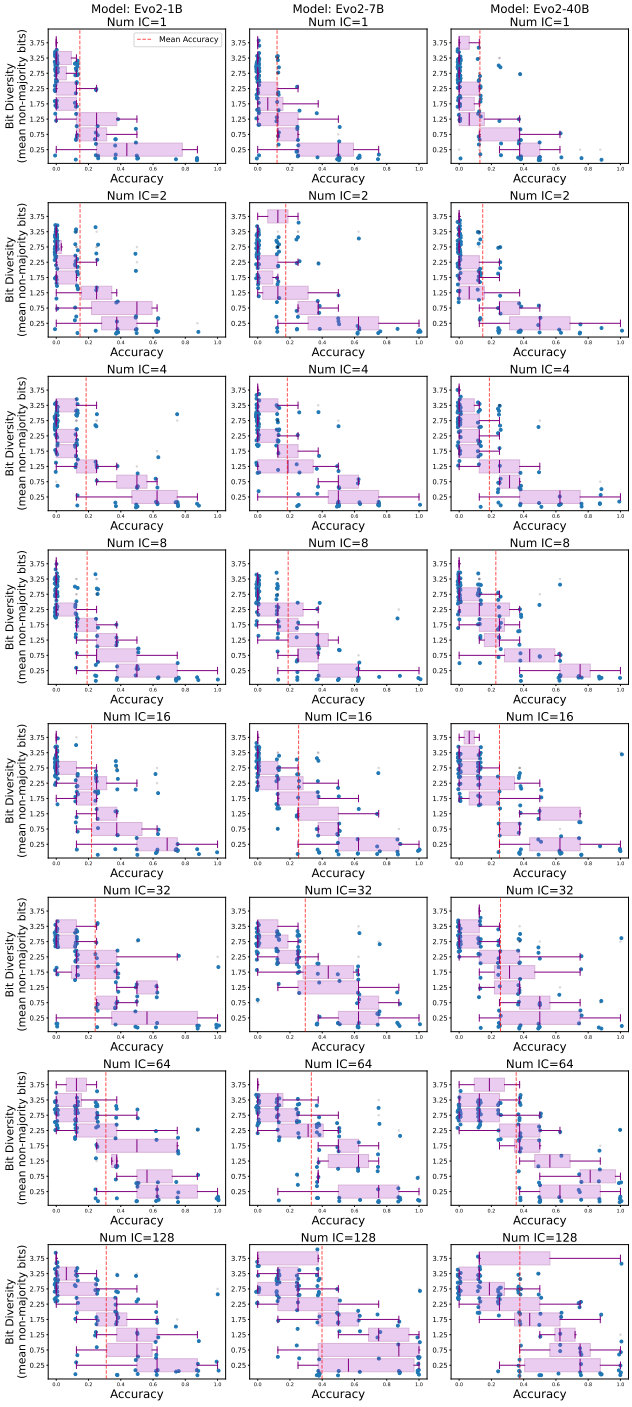


Figure 10: BitDiversity vs Accuracy for all Evo models at all ICL shot-numbers. Model performance follows a more natural pattern of increasing performance with decreasing output entropy. This highlights that it is difficult for models to decipher ICL patterns for high entropy outputs.

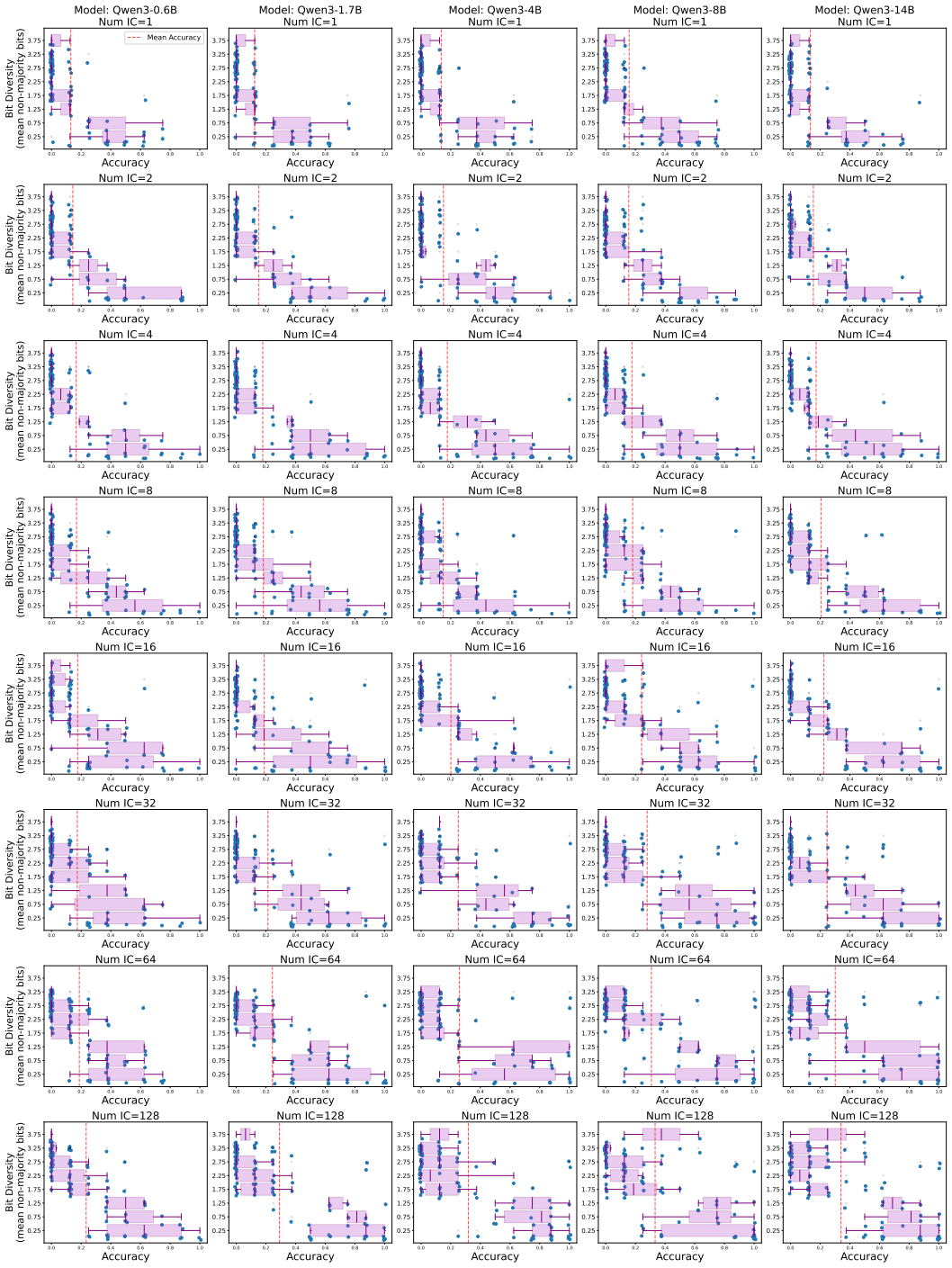


Figure 11: BitDiversity vs Accuracy for all Qwen models at all ICL shot-numbers.

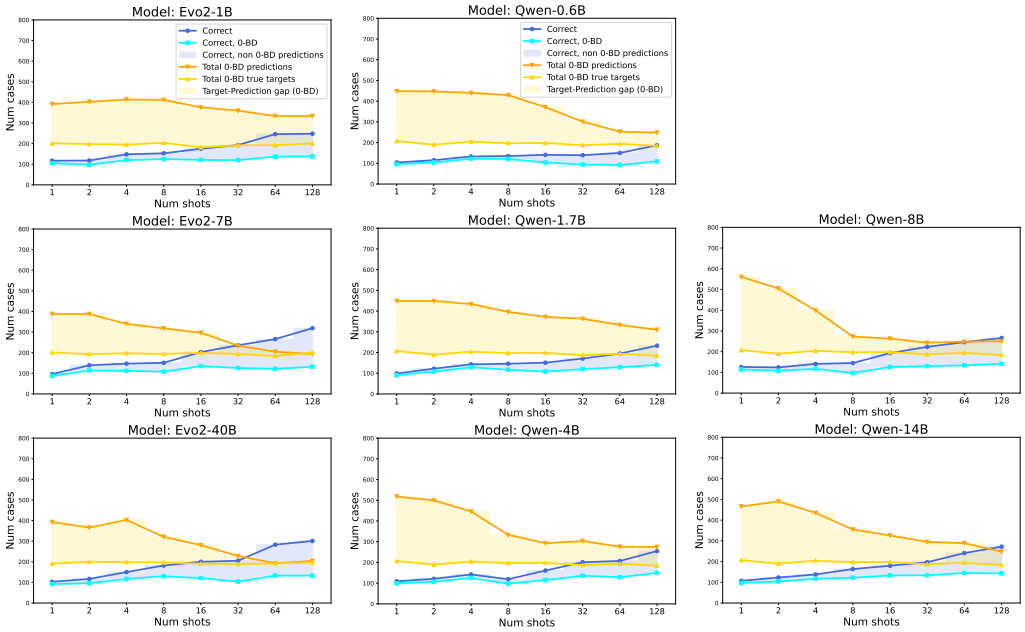


Figure 12: Enumerating cases of 0 BitDiversity. For our defined tasks, around 25% of the true targets have 0 BitDiversity4 (BD). With low-shots, models tend to produce a large number of 0-BD predictions. But this number decreases significantly with increasing shots and tends to match the actual prior value. Similarly, a majority of the baseline (1-shot) performance of models can be explained through 0-BD outputs. With a higher number of shots, the model starts learning higher entropy patterns and presents stronger evidence of ICL.