UNIVERSITY OF CALIFORNIA SAN DIEGO

Learning to Learn with Language Models

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Yufan Zhuang

Committee in charge:

Professor Jingbo Shang, Chair
Professor Taylor Berg-Kirkpatrick
Professor Zhiting Hu
Professor Julian McAuley

2026

The Dissertation of Yufan Zhuang is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2026

EPIGRAPH

A painting is never finished - it simply stops in interesting places.

Paul Gardner

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my advisor, Prof. Jingbo Shang. I met him in the late first year of my graduate studies, when we worked together on a wavelet-based attention architecture, and since then we have collaborated on many more exciting research problems. Beyond being the smartest person I know and a world-class competitive programmer, he is an exceptional mentor and research leader. He has always given me the freedom to pursue the problems that interest me most, while providing support and guidance along the way. His mentorship has shaped my research career and the way I think about problems. I feel incredibly fortunate to have had him as my PhD advisor.

I would also like to thank my long-term collaborators. To Dr. Liyuan Liu, thank you for your deep insights and support. We began working together in the latter part of my second year on several research projects, including MetaTree, Vector-ICL, and Knowledge Flow, each of which grew to have greater impact than the last. He is a wonderful mentor and a profound thinker, and I have learned so much from him. To Dr. Chandan Singh, thank you for your thoughtful guidance and support. We have been collaborating since my internship at Microsoft Research in my second year, and together we have continually pushed the frontier of how models learn from and operate on data. His deep insights into machine learning have greatly shaped my research taste.

I am grateful for the labmates and friends I have had at Shang Data Lab and at UC San Diego, Xiyuan Zhang, Ranak Roy Chowdhury, Chengyu Dong, Zilong Wang, Jiayun Zhang, Dheeraj Mekala, Zihan Wang, Zi Lin, Bill Hogan, Shuheng Li, Weitang Liu, Yangkun Wang, Xintong Li, Letian Peng, Feng Yao, Yuwei Zhang, Yiming Huang, Shang Zhou, Chenyang An and Xiaohan Fu. To all the time we had in the lab, and to all the Friday nights we spent together.

My research began during my undergraduate studies. I would like to thank Prof. Ting-Kei Peng, for giving me the first research opportunity. I would like to thank Prof. Xin Guo and Prof. Qiang Fu, for bringing me into natural language research. And I would also like to thank Prof. Baishakhi Ray, Dr. Alessandro Morari, Dr. Sahil Suneja, Dr. Yunhui Zheng and Jim Laredo

for their guidance and support during my master's studies and my time at IBM Research.

I am fortunate to have interned at Microsoft Research, Meta, AMD, Apple, and Microsoft AI. My sincere thanks go to Dr. Chandan Singh and Dr. Liyuan Liu at Microsoft Research, Dr. Xiaodong Yu at AMD, Mihai Delgeanu at Apple, and Dr. Yelong Shen at Microsoft AI. These experiences enriched my work beyond academia.

Finally, and most importantly, I would like to thank my parents and my partner for their unwavering support throughout this journey. Their love, encouragement, and belief in me gave me the strength to persevere through the darkest times.

Chapter 1, in full, is a reprint of the paper: Zhuang, Yufan, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. "Learning a Decision Tree Algorithm with Transformers." Transactions on Machine Learning Research. 2024. The dissertation author was the primary investigator and lead author of this paper.

Chapter 2, in full, is a reprint of the paper: Zhuang, Yufan, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. "Vector-ICL: In-context Learning with Continuous Vector Representations." In The Thirteenth International Conference on Learning Representations (ICLR 2025). The dissertation author was the primary investigator and lead author of this paper.

Chapter 3, in full, is a reprint of the paper: Zhuang, Yufan, Xiaodong Yu, Jialian Wu, Ximeng Sun, Ze Wang, Jiang Liu, Yusheng Su, Jingbo Shang, Zicheng Liu, and Emad Barsoum. "Self-Taught Agentic Long Context Understanding." In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5525-5537. 2025. The dissertation author was the primary investigator and lead author of this paper.

VITA

2018	Bachelor of Science with First Class Honors, Hong Kong Polytechnic University
2019	Master of Science, Columbia University
2020–2021	Research Engineer, IBM Research
2021–2022	Research Assistant, Department of Computer Science and Engineering University of California San Diego
2022–2026	Teaching Assistant, Department of Computer Science and Engineering University of California San Diego
2026	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

1. Y. Zhuang, C. Singh, L. Liu, Y. Shen, D. Zhang, J. Shang, J. Gao, and W. Chen, "Test-time Recursive Thinking: Self-Improvement without External Feedback", *arXiv 2602.03094* (2026).

2. Y. Zhuang, L. Liu, C. Singh, J. Shang, and J. Gao, "Text Generation Beyond Discrete Token Sampling", *Neural Information Processing Systems* (2025).

3. Y. Zhuang, X. Yu, J. Wu, X. Sun, Z. Wang, J. Liu, Y. Su, J. Shang, Z. Liu, and E. Barsoum, "Self-Taught Agentic Long Context Understanding", *Annual Meeting of the Association for Computational Linguistics* (2025).

4. Y. Zhuang, C. Singh, L. Liu, J. Shang, and J. Gao, "Vector-ICL: In-context Learning with Continuous Vector Representations", *International Conference on Learning Representations* (2025).

5. F. Yao*, Y. Zhuang*, Z. Sun, S. Xu, A. Kumar, and J. Shang, "Data Contamination Can Cross Language Barriers", *Empirical Methods in Natural Language Processing* (2024).

6. Y. Zhuang, L. Liu, C. Singh, J. Shang, and J. Gao, "Learning a Decision Tree Algorithm with Transformers", *Transactions on Machine Learning Research* (2024).

7. Y. Zhuang, Z. Wang, F. Tao, and J. Shang, "WavSpA: Wavelet Space Attention for Boosting Transformers' Long Sequence Learning Ability", *Neural Information Processing Systems 1st UniReps Workshop* (2023).

8. S. Suneja, Y. Zhuang, Y. Zheng, J. A. Laredo, A. Morari, and U. Khurana, "Incorporating Signal Awareness in Source Code Modeling: an Application to Vulnerability Detection", *ACM Transactions on Software Engineering and Methodology* 32, 1-40 (2023).

9. S. Suneja, Y. Zhuang, Y. Zheng, J. A. Laredo, and A. Morari, "Code Vulnerability Detection via Signal-Aware Learning", *IEEE 8th European Symposium on Security and Privacy* (2023).

10. Q. Fu, Y. Zhuang, Y. Zhu, and X. Guo, "Sleeping Lion or Sick Man? Machine Learning Approaches to Deciphering Heterogeneous Images of Chinese in North America", *Annals of the American Association of Geographers* (2022).

11. S. Suneja, Y. Zhuang, Y. Zheng, A. Morari, and J. A. Laredo, "Data-Driven AI Model Signal-Awareness Enhancement and Introspection", *arXiv preprint arXiv:2111.05827* (2021).

12. Y. Zhuang, S. Suneja, V. Thost, G. Domeniconi, A. Morari, and J. A. Laredo, "Software Vulnerability Detection via Deep Learning over Disaggregated Code Graph Representation", *arXiv preprint arXiv:2109.03341* (2021).

13. S. Suneja, Y. Zheng, Y. Zhuang, J. A. Laredo, and A. Morari, "Towards Reliable AI for Source Code Understanding", *ACM Symposium on Cloud Computing*, 403-411 (2021).

14. S. Suneja*, Y. Zheng*, Y. Zhuang*, J. A. Laredo, and A. Morari, "Probing model signal-awareness via prediction-preserving input minimization", *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2021).

15. Q. Fu, Y. Zhuang, J. Gu, Y. Zhu, and X. Guo, "Agreeing to disagree: choosing among eight topic-modeling methods", *Big Data Research* 23, 100173 (2021).

16. L. Buratti, S. Pujar, M. A. Bornea, S. McCarley, Y. Zheng, G. Rossiello, A. Morari, J. A. Laredo, V. Thost, Y. Zhuang, and G. Domeniconi, "Exploring software naturalness through neural language models", *arXiv:2006.12641* (2020).

17. S. Suneja, Y. Zheng, Y. Zhuang, J. A. Laredo, and A. Morari, "Learning to map source code to software vulnerability using code-as-a-graph", *arXiv:2006.08614* (2020).

18. Q. Fu, Y. Zhuang, J. Gu, Y. Zhu, H. Qin, and X. Guo, "Search for K: assessing five topic-modeling approaches to 120,000 Canadian articles", *BPOD Workshop at IEEE International Conference on Big Data*, 3640-3647 (2019).

ABSTRACT OF THE DISSERTATION

Learning to Learn with Language Models

by

Yufan Zhuang

Doctor of Philosophy in Computer Science

University of California San Diego, 2026

Professor Jingbo Shang, Chair

Large Language Models (LLMs) have demonstrated remarkable few-shot generalization, yet their capacity for autonomous self-improvement remains a critical frontier. This work investigates meta-learning paradigms across three fundamental dimensions: algorithmic emulation, continuous representation, and agentic self-refinement.

First, we introduce MetaTree, a transformer trained via meta-learning to generate interpretable decision tree models rather than direct predictions. By learning from both greedy and globally optimized trees, MetaTree adaptively selects its splitting strategy based on data context. This approach consistently outperforms classical decision tree algorithms on held-out datasets and successfully generalizes to deeper trees than those seen during training.

Second, we propose Vector-ICL, which extends the in-context learning capabilities of LLMs from discrete tokens to continuous vector spaces across diverse domains. Through lightweight embedding projectors trained with language modeling objectives, we enable LLMs to perform in-context learning over text, molecules, time series, graphs, and brain fMRI signals, surpassing both few-shot ICL baselines and domain-specific models.

Third, we present AgenticLU, a framework that enables LLMs to improve their own long-context understanding without relying on human annotation or stronger teacher models. Through "Chain-of-Clarifications", an agentic workflow of self-generated clarification questions and contextual grounding, combined with inference-time scaling and reinforcement learning, AgenticLU achieves significant performance gains across seven benchmarks at context windows up to 128K tokens.

Together, these works demonstrate that LLMs are versatile meta-learners: capable of learning algorithms, learning from new modalities in context, and learning to reason autonomously through self-improvement.

Introduction

Large language models can do things they were not explicitly trained to do. Given a handful of examples in a prompt, they classify, translate, reason, and solve novel problems, all without updating a single parameter [Brown et al., 2020]. They generate chain-of-thought traces that improve their own answers [Wei et al., 2022]. They transfer knowledge across tasks that share no surface-level similarity [Brown et al., 2020]. These behaviors look less like pattern matching and more like adaptation: the models appear to be learning at inference time [Olsson et al., 2022, Garg et al., 2022].

But how deep does this go? Is the flexibility of LLMs a byproduct of memorizing vast surface features [Bender et al., 2021], or does it reflect something more structured, an internalized capacity for learning itself? This question sits at the intersection of two traditions. In meta-learning, the goal has always been to build systems that acquire general strategies for adaptation, not just task-specific knowledge [Hospedales et al., 2021, Finn et al., 2017]. In the study of LLMs, the dominant framing has been in-context learning: conditioning on demonstrations in discrete text to produce outputs without weight updates [Brown et al., 2020]. This framing has been highly productive, but it captures only a narrow slice of what meta-learning can mean. It leaves open whether language models can do more than interpolate over seen demonstrations, whether they can emulate the logic of learning algorithms, whether they can generalize beyond tokenized language into continuous representations from other domains, and whether they can actively improve their own reasoning rather than passively conditioning on context.

This dissertation argues that they can. Specifically, it develops and tests the thesis that large language models are not only statistical learners but flexible computational systems capable

of internalizing procedures, operating over heterogeneous forms of information, and improving through iterative self-directed reasoning. We study this claim through three complementary dimensions of meta-learning. The first is algorithmic emulation: can a neural model meta-learn to execute explicit learning algorithms in an interpretable form? The second is continuous representation: can the in-context learning capacity of LLMs extend beyond discrete tokens to continuous vector representations from arbitrary domains? The third is agentic self-refinement: can a language model improve its own performance by autonomously generating and refining intermediate reasoning processes? Rather than treating few-shot prompting, multimodal adaptation, and self-improvement as separate phenomena, this thesis argues they are manifestations of a single underlying capacity: general-purpose meta-learning behavior emerging in large language models. The remainder of this chapter summarizes the three lines of work that develop this argument and then discusses their collective implications. Chapter 1 presents MetaTree [Zhuang et al., 2024a] (algorithmic emulation), Chapter 2 presents Vector-ICL [Zhuang et al., 2024b] (continuous representation), and Chapter 3 presents AgenticLU [Zhuang et al., 2025] (agentic self-refinement).

The most direct test of whether a model has internalized meta-learning is to ask it to produce a learning algorithm, not just a prediction. If a model can take a dataset as input and output an effective, interpretable predictor, choosing different construction strategies depending on the structure of the data, then it has learned something about the process of learning itself. MetaTree realizes this idea for decision trees. We train a transformer to generate decision tree models rather than direct predictions, framing tree construction as a sequence generation problem conditioned on a dataset context. Unlike classical methods that commit to a single induction strategy such as greedy splitting, global optimization, or a fixed heuristic, MetaTree is trained on trees produced by multiple strategies and learns to select among them based on the input. The result is a model that treats algorithm design as a meta-learning problem: given a new dataset, it infers what kind of tree to build. MetaTree outperforms classical decision tree baselines on held-out datasets and, notably, generalizes to deeper tree structures than those seen during training.

This generalization provides evidence that MetaTree has captured transferable principles of when different splitting strategies are appropriate, rather than memorizing tree-building recipes. More broadly, it establishes that neural sequence models can internalize procedural knowledge about learning, not just declarative knowledge about data.

This finding raises a natural follow-up question. If a language model can internalize the logic of a learning algorithm over structured tabular data, can its in-context learning ability, usually studied over discrete text, extend to other kinds of inputs entirely? Many real-world domains, including molecular structures, neural signals, and temporal dynamics, are most naturally represented as continuous vectors rather than discrete tokens. If the meta-learning capacity of LLMs is truly general, it should not be confined to the symbolic medium the model was trained on. Vector-ICL tests this hypothesis by equipping language models with lightweight embedding projectors trained under language modeling objectives, enabling the model to consume and reason over continuous inputs while preserving its in-context learning machinery. The key design choice is to align the projection with the model's existing representational structure rather than fine-tuning for each new domain, keeping the general-purpose adaptation mechanism intact. We evaluate across text embeddings, molecular representations, time series, graph structures, and brain fMRI signals. In each setting, Vector-ICL surpasses conventional few-shot prompting and, in many cases, outperforms domain-specialized models. These results demonstrate that the meta-learning capacity of LLMs extends to continuous, heterogeneous representations when the interface between modality and model is properly constructed.

Yet extending in-context learning to new modalities also exposes a limitation of passive conditioning. Whether the demonstrations are textual or continuous, simply presenting more examples works well for straightforward tasks but breaks down when the reasoning required becomes complex. The model needs to do something more active: construct intermediate reasoning that helps it make sense of what it has seen. The first two contributions establish that language models can internalize learning algorithms and apply in-context learning across modalities, but both treat the model as a passive processor that receives context and produces

output. A stronger form of meta-learning would involve the model actively shaping its own reasoning, identifying what it does not understand, generating intermediate steps to resolve confusion, and using those steps to improve downstream performance. AgenticLU instantiates this idea for long-context understanding. Rather than relying on human annotations or distillation from stronger teacher models, the model generates its own clarification questions about the input, grounds them in the original context, and uses these self-generated intermediate steps to improve comprehension through a process we call Chain-of-Clarifications. The framework is further strengthened through inference-time scaling and reinforcement learning, enabling the system to benefit from additional computation and self-produced training signals. Across seven benchmarks and context windows up to 128K tokens, AgenticLU achieves substantial gains over strong baselines, including models trained with pure long-context objectives. Instead of treating long-context understanding as a passive retrieval problem, the model actively constructs reasoning scaffolds that improve with more inference-time compute and can be refined through RL without external supervision.

Together, these three lines of work close the loop on the dissertation's argument. MetaTree shows that language models can produce learning algorithms. Vector-ICL shows they can learn from context across heterogeneous domains. AgenticLU shows they can improve their own reasoning by generating and refining intermediate chain-of-thought. The common thread is that large neural models can acquire mechanisms for adaptation more general than task-specific pattern matching. They can infer procedures, reinterpret inputs, and construct intermediate computations that enable more effective generalization.

Chapter 1

Learning a Decision Tree Algorithm with Transformers

Decision trees are renowned for their ability to achieve high predictive performance while remaining interpretable, especially on tabular data. Traditionally, they are constructed through recursive algorithms, where they partition the data at every node in a tree. However, identifying a good partition is challenging, as decision trees optimized for local segments may not yield global generalization. To address this, we introduce MetaTree, a transformer-based model trained via meta-learning to directly produce strong decision trees. Specifically, we fit both greedy decision trees and globally optimized decision trees on a large number of datasets, and train MetaTree to produce only the trees that achieve strong generalization performance. This training enables MetaTree to emulate these algorithms and intelligently adapt its strategy according to the context, thereby achieving superior generalization performance.[1]

1.1 Introduction

Transformers [Vaswani et al., 2017] have demonstrated the capacity to generate accurate predictions on tasks previously deemed impossible [OpenAI, 2023, Betker et al., 2023]. Despite this success, it remains unclear if they *can directly produce models* rather than predictions. In this work, we investigate whether transformers can generate binary decision trees and explore

[1] Code available at: https://github.com/EvanZhuang/MetaTree.

the mechanisms by which they do so. We select decision trees as they are foundational building blocks of modern machine learning and hierarchical reasoning. They achieve state-of-the-art performance across a wide range of practical applications [Grinsztajn et al., 2022] and additionally offer interpretability, often a critical consideration in high-stakes situations [Rudin, 2018, Murdoch et al., 2019].

Decision trees are traditionally constructed using algorithms based on greedy heuristics [Breiman et al., 1984, Quinlan, 1986]. To overcome the bias imposed by greedy algorithms, recent work has proposed global, discrete optimization methods for fitting decision trees [Lin et al., 2020, Hu et al., 2019, Bertsimas and Dunn, 2017]. While these approaches have been effective in tabular contexts, a significant challenge lies in their non-differentiability, which raises difficulties when integrating them into deep learning models. Moreover, full decision tree optimization is NP-hard [Laurent and Rivest, 1976], rendering it practically infeasible to compute optimal trees with large tree depths.

In this work, we introduce MetaTree, a transformer-based model designed to construct a decision tree on a tabular dataset. MetaTree recursively performs inference to decide the splitting feature and value for each decision node (Fig. 1.1a). We train MetaTree to produce high-performing decision trees. Specifically, we fit both greedy decision trees and optimized decision trees on a large number of datasets. We then train MetaTree to generate the trees that demonstrated superior generalization performance (Fig. 1.1c). This training strategy endows MetaTree with a unique advantage: it learns not just to emulate the underlying decision tree algorithm, but also to implicitly select which algorithmic approach is more effective for a particular dataset. This adaptability adds significant flexibility to traditional methods that are bound to a single algorithmic framework. MetaTree's architecture leverages an alternating row and column attention mechanism along with a learnable absolute positional bias for the tabular representations (Fig. 1.1b).

MetaTree produces highly accurate trees on a large set of real-world datasets that it did not see during training, consistently outperforming traditional decision tree algorithms (Table 1.1 and

Table 1.1. Comparing each algorithm's single-tree performance over the 91 held-out datasets (91D) and the 13 Tree-of-prompts datasets (ToP), among tree depth 2, 3, and 4. MetaTree generalizes well, despite only being trained on depth-2 trees. Listed are the average rank with standard deviation over the datasets. We also report the number of times an algorithm is in the first place (champion). †GOSDT encounters out-of-memory (OOM) errors for tree depths greater than 2 due to the NP-hard complexity of solving for the optimal tree. We report the memory usage for GOSDT in Table 1.2.

Config		**MetaTree**	GOSDT	CART	ID3	C4.5
91D, depth-2	avg. rank	$\mathbf{1.34 \pm 0.84}$	2.68 ± 1.20	3.30 ± 0.90	3.30 ± 0.92	4.36 ± 1.15
	champion	**72**	13	2	2	2
ToP, depth-2	avg. rank	$\mathbf{1.08 \pm 0.27}$	2.77 ± 1.48	3.00 ± 1.18	2.77 ± 1.25	3.38 ± 1.60
	champion	**7**	3	1	2	0
91D, depth-3	avg. rank	$\mathbf{1.25 \pm 0.64}$	OOM†	2.65 ± 0.78	2.57 ± 0.80	3.53 ± 0.84
	champion	**76**	-	5	8	2
ToP, depth-3	avg. rank	$\mathbf{1.31 \pm 0.82}$	OOM†	2.54 ± 1.15	2.46 ± 1.08	2.69 ± 0.99
	champion	**9**	-	2	2	0
91D, depth-4	avg. rank	$\mathbf{1.25 \pm 0.67}$	OOM†	2.79 ± 0.79	2.51 ± 0.76	3.44 ± 0.92
	champion	**76**	-	6	5	4
ToP, depth-4	avg. rank	$\mathbf{1.15 \pm 0.36}$	OOM†	2.46 ± 0.93	2.46 ± 1.01	2.69 ± 1.14
	champion	**9**	-	1	2	1

more in Sec. 1.5). Further analysis shows that MetaTree's performance improvement comes from its ability to dynamically switch between a greedy or global approach depending on the context of the split (Sec. 1.6.1). Finally, a bias-variance analysis shows that MetaTree successfully achieves lower empirical variance than traditional decision-tree algorithms (Sec. 1.6.2).

1.2 Related work

Decision trees

There is a long history of greedy methods for fitting decision trees, e.g., CART [Breiman et al., 1984], ID3 [Quinlan, 1986], or C4.5 [Quinlan, 2014]. Recent work has explored fitting trees overcoming the limitations of greedy methods by using global optimization methods [Lin et al., 2020, Hu et al., 2019, Bertsimas and Dunn, 2017, Jo et al., 2023]. These methods can

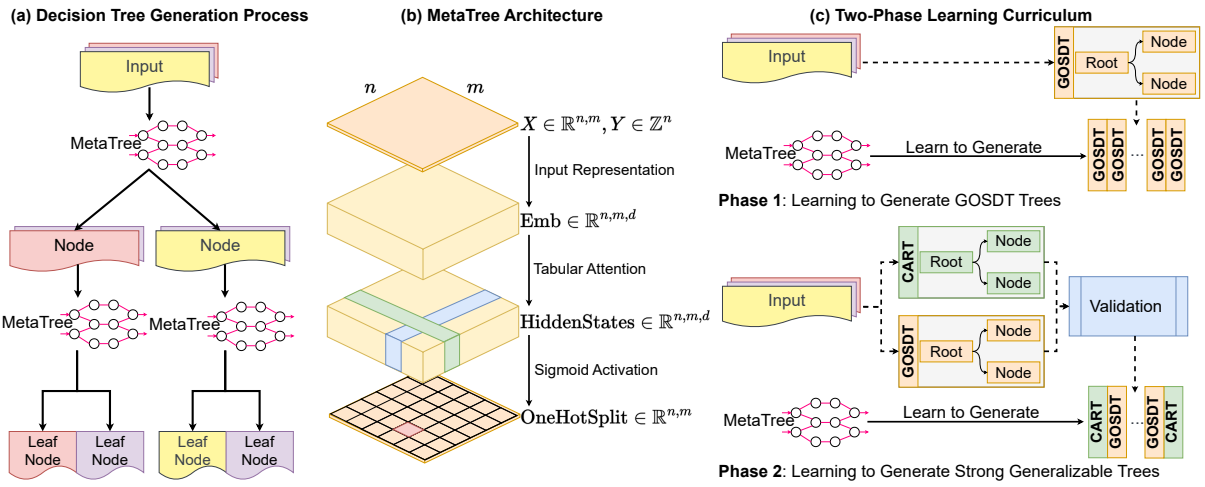


Figure 1.1. MetaTree Methodology. The creation of a decision tree, depicted in (a), entails recursive MetaTree calls. MetaTree only assesses the current state for its decision-making. (b) shows MetaTree's architecture, in which the tabular input (n data points of m features) is embedded in a representation space, processed with row and column attention at each layer, and the output is a one-hot mask indicating the splitting feature j and threshold $X_{i,j}$. (c) illustrates MetaTree's two-phase learning curriculum: in the first phase, the focus is exclusively on learning from the optimized GOSDT trees, to closely emulate the behavior of GOSDT algorithm. Then in the second phase, the training process incorporates data from both the GOSDT and CART trees, generating the ones that have better generalization capabilities.

improve performance given a fixed tree size, but often incur a prohibitively high computational cost. Other recent studies have improved trees through regularization [Agarwal et al., 2022, Chipman and McCulloch, 2000, Yıldız and Alpaydın, 2013], iterative updates [Carreira-Perpinán and Tavallali, 2018, Hada et al., 2023], or increased flexibility [Grossmann, 2004, Valdes et al., 2016, Tan et al., 2022, Sorokina et al., 2007, Hiabu et al., 2020].

Tree-based models maintain state-of-the-art prediction performance across most tabular applications [Grinsztajn et al., 2022, Kornblith et al., 2022], especially when used in ensembles such as Random Forest [Breiman, 2001], gradient-boosted trees [Freund et al., 1996, Chen and Guestrin, 2016], and BART [Chipman et al., 2010]. Some works have sought to combine these tree-based models with deep-learning based models to improve predictive performance when fitting a single dataset [Frosst and Hinton, 2017, Zantedeschi et al., 2021, Tanno et al., 2019, Lee

and Jaakkola, 2019, Biau et al., 2019].

Recent research has increasingly focused on exploring the synergy between tree-based models and large language models (LLMs). One area of investigation involves leveraging trees to guide and improve the generation processes of LLMs. Using tree structure in searching for prompts provides a hierarchical framework that structures the prompt space more effectively. It can enhance the coherence and logical flow of generated text [Yao et al., 2023], and improve task performance without needing to finetune the LLM [Morris et al., 2023]. LLMs can also be used to build stronger decision trees. Aug-imodels [Singh et al., 2023] has demonstrated that integrating LLMs during the training phase of decision trees can significantly enhance their performance and interpretability.

Learning models/algorithms

Some learning-based methods have focused on improving algorithms, largely based on combining transformers with deep reinforcement learning, e.g. faster matrix multiplication [Fawzi et al., 2022] or faster sorting algorithms [Mankowitz et al., 2023]. One new work studies using LLMs to iteratively generate and refine code to discover improved solutions to problems in computer science and mathematics [Romera-Paredes et al., 2023].

Other works focus on transformers' ability to learn in context. Zhou et al. [2023] probe simple tasks for length generalization and find that transformers can generalize to a certain class of problems easily. One very related work studies whether Transformers can successfully learn to generate predictions from linear functions, and even decision trees and small MLPs in context [Garg et al., 2022]. Despite these successes, recent works also find limitations in transformers' ability to generalize in certain contexts, e.g. to distribution shifts [Yadlowsky et al., 2023] or to arithmetic changes [Dziri et al., 2023].

Meta-learning

Often referred to as "learning-to-learn", meta-learning has gained increasing attention in recent years [Finn et al., 2017, Nichol et al., 2018, Hospedales et al., 2021]. Meta-learning

algorithms are designed to learn a model that works well on an unseen dataset/task given many instances of datasets/tasks [Vilalta and Drissi, 2002]. Particularly related to MetaTree are works that apply meta-learning with transformers to tabular data [Hollmann et al., 2022, Feuer et al., 2023, Onishi et al., 2023, Hegselmann et al., 2023, Gorishniy et al., 2023, Manikandan et al., 2023, Zhu et al., 2023]. Our work follows the direction of meta-learning with a focus on directly outputting a model informed by the insights of established algorithms.

1.3 Method: MetaTree

Problem definition

When generating a decision tree, we are given a dataset $D = (x_i, y_i)_{i=1}^n$ where $x_i \in \mathbb{R}^m$ represents the input features and $y_i \in \{1, \ldots, K\}$ corresponds to the label for each instance. At each node, a decision tree identifies a split consisting of a feature $j \in \{1, \ldots, m\}$ and a threshold value $v \in \mathbb{R}$, such that D can be partitioned into two subsets by thresholding the value of the j^{th} feature. The dataset is partitioned recursively until meeting a pre-specified stopping criterion, here a maximum tree depth. To generate a prediction from the tree, a point is passed through the tree until reaching a leaf node, where the predicted label is the majority label of training points falling into that leaf node. We illustrate MetaTree's generation pipeline, architecture design and its two-phase learning curriculum in Fig. 1.1.

1.3.1 Generating a decision tree model: representation and model design

Decision trees are typically fit using a top-down greedy algorithm such as CART [Breiman et al., 1984]. These algorithms greedily select the split at each node based on a criterion such as the Gini impurity. This class of methods, while efficient, often results in sub-optimal solutions. More recent work has studied the generation of "optimal" decision trees seeking a solution that maximizes predictive performance subject to minimizing the total number of splits in a tree. This can be formulated as a tree search, in which recursive revisions on the tree happen when all children of a search node are proven to be non-optimal [Lin et al., 2020]. However, finding

optimal trees is intractable for even modest tree depths, and can also quickly overfit to noisy data.

Model design: divide and conquer with learned speculative planning

Our approach, MetaTree, aims to bridge the gap between existing methods by generating highly predictive decision trees, as illustrated in Fig. 1.1(a). To construct a single tree, MetaTree is recursively called at each node. The process begins by presenting the entire dataset to the model, in which MetaTree creates the first split (i.e. a threshold value on a feature dimension). The dataset is then filtered into two subsets by the root node's split, and each subset (i.e., left child and right child) is individually passed through the model, while masking out the opposite subset. This recursive process continues until a predetermined maximum depth for the tree is reached. Although MetaTree only outputs one split at a time, its ability to consider the entire dataset when making the initial split and utilize multiple Transformer layers enables it to make adaptive, non-greedy splits. We show experimental results supporting this claim in Sec. 1.6.1 and we include related case studies in Sec. 1.6.5.

Representing numerical inputs: learnable projection and positional bias

MetaTree takes matrices of real-valued numbers as input. We use a multiplicative embedding to project all the numerical features into embedding space and add the class embedding onto it. The aggregated embedding is then transformed via a two-layer MLP. Specifically, given a n-row m-column input $X \in \mathbb{R}^{n,m}$ and its k-class label $Y \in \{1,\ldots,k\}^n$, with Y_{oh} denotes Y in

one-hot format, the embedding is computed as follows:

$$\text{(Feature Embedding)} \quad \text{Emb}_{\text{x}}(X) = X \otimes W_x \in \mathbb{R}^{n,m,d}, \quad W_x \in \mathbb{R}^d \tag{1.1}$$

$$\text{(Label Embedding)} \quad \text{Emb}_{\text{y}}(Y) = Y_{oh} \cdot W_y \in \mathbb{R}^{n,d}, \quad W_y \in \mathbb{R}^{k,d} \tag{1.2}$$

$$\text{(Position Embedding)} \quad B_{(i,j,k)} = b_{1(j,k)} + b_{2(i,k)} \tag{1.3}$$

$$\text{(Final Representation)} \quad \text{Emb} = \text{MLP}(\text{Emb}_{\text{x}}(X) + \text{Emb}_{\text{y}}(Y) + B) \tag{1.4}$$

$$\text{Emb} \in \mathbb{R}^{n,m,d}, B \in \mathbb{R}^{n,m,d}, b_1 \in \mathbb{R}^{m,d}, \; b_2 \in \mathbb{R}^{n,d}$$

For each number in the matrix X, it is transformed into $\mathbb{R}^d$ space via multiplication with W_x, then added to the class embedding of Y. The final embedding is obtained by putting the aggregated embedding plus the positional bias terms b_1, b_2 through an MLP.

We normalize each feature dimension per batch to have mean of 0 and variance of 1. Prior to inference, we also add a truncated Gaussian noise to categorical features; this improves performance on discrete features since the model is mostly trained on continuous data.

Tabular self-attention: row and column-wise information processing

Since our tabular input shares information across rows and columns, we apply attention to both the row dimension and column dimension in each Transformer layer. For l^{th} layer, given input in hidden space $X_h^{(l)} \in \mathbb{R}^{n,m,d}$, the output of the tabular attention $Y_h^{(l)} \in \mathbb{R}^{n,m,d}$ is computed as:

$$\text{ColAttn}(X_h^{(l)}) = \text{Softmax}\left(Q_{\text{col}}^{(l)\top} K_{\text{col}}^{(l)}\right) V_{\text{col}}^{(l)} \tag{1.5}$$

$$\text{RowAttn}(X_h^{(l)}) = \text{Softmax}\left(Q_{\text{row}}^{(l)\top} K_{\text{row}}^{(l)}\right) V_{\text{row}}^{(l)} \tag{1.6}$$

$$Y_h^{(l)} = \text{ColAttn}(X_h^{(l)}) + \text{RowAttn}(X_h^{(l)}) + X_h^{(l)} \tag{1.7}$$

where $X_h^{(0)} = \text{Emb}(\text{X}, \text{Y})$ for the first layer, $X_h^{(l)} = \text{MLP}(Y_h^{(l-1)})$ for the rest the layers; for the column/row query Q, key K, value V projections, $X_h^{(l)} \in \mathbb{R}^{n,m,d}$ is first reshaped to $X_{h,\text{col}}^{(l)} \in \mathbb{R}^{n,m,d}$ and $X_{h,\text{row}}^{(l)} \in \mathbb{R}^{m,n,d}$ where the Softmax is conducted on the corresponding column/row dimension, *i.e.* second last dimension. The Q, K, V are constructed in the following manner, we will explain this for column attention only where the row attention follows the same design: $Q_{\text{col}}^{(l)} = W_{Q,\text{col}}^{(l)} X_{h,\text{col}}^{(l)}$, $K_{\text{col}}^{(l)} = W_{K,\text{col}}^{(l)} X_{h,\text{col}}^{(l)}$, $V_{\text{col}}^{(l)} = W_{V,\text{col}}^{(l)} X_{h,\text{col}}^{(l)}$, where the linear projections $W_{Q,\text{col}}^{(l)}, K_{\text{col}}^{(l)}, V_{\text{col}}^{(l)} \in \mathbb{R}^{d,d}$ are all learnable parameters. The dimension orders will be shuffled back after column/row attention such that $\text{ColAttn}(X_h^{(l)}), \text{RowAttn}(X_h^{(l)}) \in \mathbb{R}^{n,m,d}$.

Attention is being applied in the row and column dimensions individually, it will first gather information over n rows and then over m columns with an $O(n^2 + m^2)$ complexity. This alleviates the computing cost compared to reshaping the table as a long sequence, which would require an $O\left(n^2 m^2\right)$ complexity, while effectively gathering and propagating information for the entire table.

1.3.2 Training objective: cross entropy with Gaussian smoothing

The main task of our model is to select a feature and value to split the input data. This process involves selecting a specific element from the input matrix $X \in \mathbb{R}^{n,m}$. Suppose the choice is made for $X_{i,j}$, it is equivalent to a decision that splits the data along the j^{th} feature with value $X_{i,j}$. Our design for the model's output and the corresponding loss function is based on this fundamental principle of split selection. The model's output is passed through a linear projection to scale down from $\mathbb{R}^{n,m,d}$ to $\mathbb{R}^{n,m,1}$, and the final output is obtained after a Sigmoid activation.

We train our model with supervised learning. In the ground truth tree, each node contains the feature index and the splitting value. This is equivalent to a one-hot mask over the input table where the optimal choice of feature and value is marked as one and the rest marked as zero. However, directly taking this mask as the training signal raises issues: some data points might have similar or exactly the same values along the same feature as the optimal split, and masking out these data points would deeply confuse the model. Hence we use a Gaussian smoothing over

this mask as our loss target based on the value distance over the chosen feature. We denote the ground truth splitting feature and value as j^* and v^*, the training target M:

$$M = \begin{cases} \exp -\frac{(X[:,j]-v^*)^2}{2\sigma^2}, & \text{if } j = j^* \\ 0, & \text{if } j \neq j^* \end{cases} \tag{1.8}$$

where σ is a hyperparameter, controlling the smoothing radius. We use Binary Cross Entropy (BCE) to calculate the loss between our model output and the training target M.

Learning curriculum: learning from optimized trees to best generalizing trees

Our training approach is tailored to accommodate the mixed learning signals derived from the two distinct algorithms, each with its unique objectives and behaviors. The task is difficult. Mimicking the optimization-based decision tree algorithm is already challenging (i.e. approximating solutions for an NP-hard problem), but MetaTree must also learn to generate the split that has the better generalization potential out of the two (CART v.s. GOSDT).

To effectively train our model, we use a learning curriculum in our experiments (as illustrated in Fig. 1.1c. In the first phase, the focus is exclusively on learning from the optimization-based GOSDT trees. The goal during this stage is to closely emulate the behavior of GOSDT algorithm. Then in the second phase, the training process incorporates data from both the GOSDT and CART trees (see details in Sec. 1.4.1) to train MetaTree. This two-stage approach enables our model to assimilate the characteristics of both algorithms, facilitating better generalization capabilities. (See Sec. 1.6.5 for ablation studies on the training curriculum.)

We selected CART and GOSDT as they are representative and effective algorithms in the classes of greedy and optimized decision tree algorithms. As shown in our experiments (Fig. 1.2), they are consistently first and second amongst classical algorithms. Training with the best tree out of an algorithm ensemble could potentially further improve performance, but for the simplicity of our analyses we leave that for future explorations.

MetaTree in inference: generating decision tree models

Note that MetaTree is equivalent to an algorithm, the inference process is slightly different from typical deep learning models. When deploying MetaTree on an unseen dataset $(X_{\text{train}}, Y_{\text{train}}, X_{\text{test}}, Y_{\text{test}})$, we will first generate decision trees from the train set $(X_{\text{train}}, Y_{\text{train}})$, then use the generated trees to make predictions Y_{pred} on X_{test}:

$$\text{DecisionTreeModel} = \text{MetaTree}(X_{\text{train}}, Y_{\text{train}})$$
$$Y_{\text{pred}} = \text{DecisionTreeModel}(X_{\text{test}})$$

where DecisionTreeModel is constructed by recursively calling MetaTree, as shown in Fig. 1.1(a).

1.4 Experimental setup

1.4.1 Datasets

We use 632 classification datasets from OpenML [Vanschoren et al., 2013], Penn Machine Learning Benchmarks [Romano et al., 2021], along with a synthetic XOR dataset. We require each dataset to have at least 1000 data points, at most 256 features, at most 10 classes, and less than 100 categorical features, with no missing data. We randomly select 91 datasets as the left-out test set for evaluating our model's generalization capability while making sure they and their variants do not appear in the training set.

We generate our decision-tree training dataset in the following manner: for each dataset, we first divide it into train and test sets with a 70:30 split; then we sample 256 data points with 10 randomly selected feature dimensions from the training set and fit a GOSDT tree [Lin et al., 2020] and a CART tree [Breiman et al., 1984] of depth 2 both; we later record the accuracy of the two trees on the test set; We repeat this process and generate 10k trees for each dataset. In total, we have 10,820,000 trees generated for training.

We have both GOSDT and CART trees generated for all these datasets; for clarity, we refer to the GOSDT trees as the *GOSDT dataset*, the CART trees as the *CART dataset*, and the trees that have the best evaluation accuracy on their respective test set between CART and GOSDT as the *GOSDT+CART dataset*.

We generate a synthetic XOR dataset with the following algorithm: we first randomly sample 256 data points in the 2-dimensional bounding box $\{x|x \in [-1,1]^2\}$; then randomly generate the ground truth XOR splits inside the bounding box depending on the pre-specified level (for example, level 1 XOR has 3 splits, level 2 XOR has 15 splits, and the root node split can take place randomly $\in [-1,1]$ while the rest of the split is sampled randomly inside the dissected bounding boxes, see examples in Fig. 1.5) and assign the class labels according to the splits; at the final step, we add in label flipping noise and additional noisy feature dimensions consisting of uniform noise within $[-1,1]$.

We additionally test MetaTree on the 13 Tree-of-prompt datasets from Morris et al. [2023]. They are tabular datasets constructed from text classification tasks; the input X is the LLM's response to a set of prompts accompanying the text (yes or no), and the output Y is the class label. Successfully building trees on these datasets shows the potential for MetaTree to help in steering large language models. Moreover, since MetaTree is differentiable, it could be integrated into the training process of some LLMs.

1.4.2 Baselines

We use GOSDT [Lin et al., 2020], CART [Breiman et al., 1984], ID3 [Quinlan, 1986] and C4.5 [Quinlan, 2014] as our baselines, as representative optimal and greedy decision tree algorithms. For GOSDT, we utilize the official implementation, using gradient-boosted decision trees for the initial label warm-up with the number of estimators equal to 128, a regularization factor of 1e-3. For CART and ID3, we use the sklearn implementation [Pedregosa et al., 2011], setting the splitting criterion as Gini impurity and entropy respectively. For C4.5, we use the imodels implementation [Singh et al., 2021] with the default setting.

1.4.3 Model configurations

We use the LLaMA [Touvron et al., 2023a] architecture as the base Transformer. For MetaTree, we set the number of layers as 12, the number of heads as 12, the embedding dimension as 768, MLP dimension as 3072. We can use a maximum of 256 data points and 10 features for producing a single tree, as Transformers' memory usage and runtime would grow quadratically with the sequence length, see our discussion on this in Sec. 1.7.

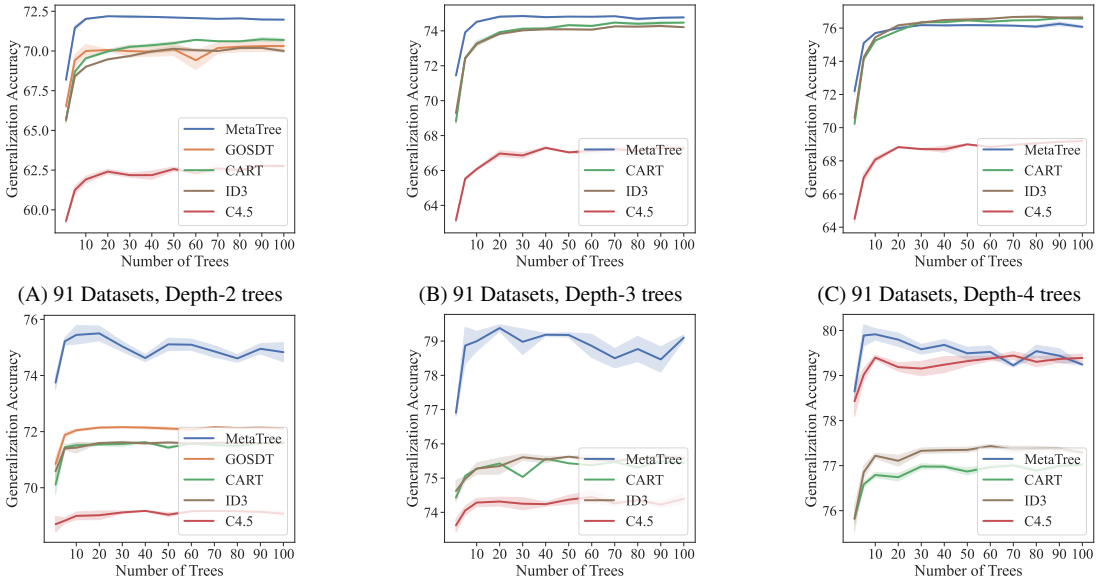
We pretrain our model from scratch on the *GOSDT dataset*, and after training converges, we finetune it on the *GOSDT+CART dataset*. This curriculum improves performance compared to direct training on the *GOSDT+CART dataset* (as shown in Sec. 1.6.5). We also show an ablation with RL training objective in Sec. 1.6.5. Detailed hyperparameters are shown in Sec. A.1.

1.5 Results: On the Generalization Ability of MetaTree

In this section, we present MetaTree's performance on real-world datasets previously unseen by the model (in Table 1.1 and Fig. 1.2). We compare it with established algorithms, GOSDT, CART, ID3, C4.5, and find that it performs favorably when used as a single tree and as an ensemble. We focus on three questions: (1) Can MetaTree effectively generalize to real-world data it has not encountered before? (2) Is MetaTree capable of generating decision trees deeper than those it was trained on?, and (3) Can MetaTree be used in an LLM setting to accurately steer model outputs?

Generalizing to new datasets: Fig. 1.2A

To address the first question, we rigorously evaluate MetaTree across 91 datasets that were excluded from its training. For each dataset, we adopt the standard 70/30 split to create the train and test sets, then repeatedly sample from the train set and run the decision tree algorithms (MetaTree, GOSDT, CART, ID3, or C4.5) to form tree ensembles with specified number of trees from $\{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. The majority prediction across trees is taken as the ensemble prediction and accuracy is averaged across all datasets. The entire evaluation



(A) 91 Datasets, Depth-2 trees (B) 91 Datasets, Depth-3 trees (C) 91 Datasets, Depth-4 trees

(D) Tree-of-prompt Datasets, Depth-2 trees (E) Tree-of-prompt Datasets, Depth-3 trees (F) Tree-of-prompt Datasets, Depth-4 trees

Figure 1.2. MetaTree demonstrates strong generalization on real-world datasets. MetaTree generalizes well to 91 held-out datasets for (A) depth-2 trees (B) depth-3 trees and (C) depth-4 trees, despite only being trained to produce depth-2 trees. MetaTree also generalizes well to the 13 Tree-of-prompts datasets for (D) depth-2 trees (E) depth-3 trees (F) depth-4 trees, which requires constructing a tree to steer a large language model [Morris et al., 2023]. Each plot shows the average test accuracy for tree ensembles of size $\{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, with error bars indicating the standard deviation.

process is replicated across 5 independent runs and the standard deviation is shown as error bars in the plot.

The result shows that MetaTree demonstrates a consistent and significant performance advantage over the baseline algorithms. GOSDT outperforms CART and the other greedy algorithms when the number of trees is small; this corresponds well to GOSDT's tendency to overfit on training data as it is designed to optimally solve for the train set. It also brings a higher variance in GOSDT's generalization performance. CART and ID3 exhibit similar performance due to their algorithmic similarities. In contrast, C4.5 underperforms, likely due to excessive node pruning which hampers performance when model complexity is not high enough.

In addition to the above results, we provide the average single-tree fitting accuracy of the algorithms in Sec. 1.6.5 for a more comprehensive view of their performance.

Generalizing to deeper trees: Fig. 1.2B,C

Going beyond MetaTree's capability to generalize to new data, we now study whether MetaTree can generate deeper trees. To answer this question, we ask MetaTree to generate trees with depth 3 and 4, and compare its generalization performance with the baseline algorithms, similar to the aforementioned evaluation process. [2]

The result is shown in Fig. 1.2B,C. It can be observed that MetaTree still consistently outperforms the baseline algorithms at depth 3. At depth 4, MetaTree outperforms the baselines in the single-tree scenario or when the number of trees is small, and the greedy algorithms (CART, ID3) catch up with the growing ensemble size.

One reason MetaTree can generalize to deeper trees is that it is designed to generate splitting decisions at each node, hence the generated tree is not dependent on the tree depth. Besides, the model has trained on depth 2 trees, i.e. the root node split and two children split, we believe MetaTree might have learned to behave as an induction algorithm, thereby generating deeper trees with high quality.

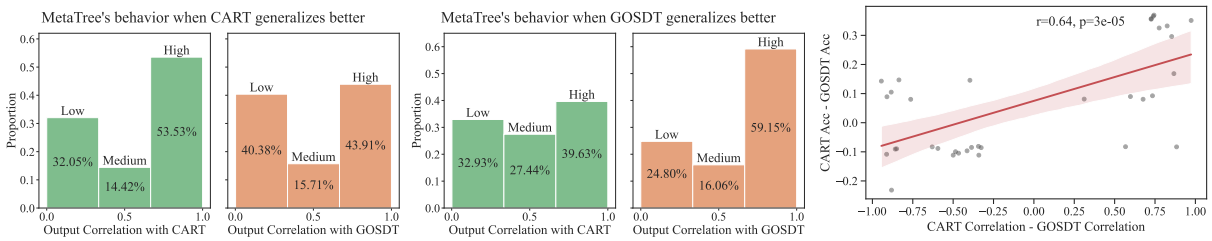
Tree of prompts dataset: Fig. 1.2D,E,F

We evaluate MetaTree on the 13 Tree-of-prompt datasets, that consist of purely categorical features, with the input being LLM's answer to a set of prompts (yes or no) and output being the classification label of the text. Fig. 1.2D,E,F again compares MetaTree to GOSDT, CART, ID3, and C4.5. MetaTree maintains a higher level of generalization accuracy across almost all tree counts when compared to GOSDT, CART, ID3 and C4.5.

GOSDT, CART, and ID3 show lower generalization accuracies, with GOSDT performing slightly better than CART and ID3. Notably, C4.5's performance jumps at depth-4 and surpasses CART and ID3, potentially due to its node pruning being effective at deeper tree depth.

This evaluation demonstrates MetaTree's great performance on Tree-of-prompt datasets,

[2]Note that GOSDT can not stably generate trees with a depth greater than 2 without incurring Out-of-Memory or Out-of-Time errors on machines with up to 125G memory (see Sec. 1.6.3 for memory usage analysis). Hence GOSDT is excluded from this study.



(a) **(b)** **(c)**

Figure 1.3. We show that MetaTree learns to adapt its splitting strategy for better generalization. (a), (b) The figures demonstrate MetaTree's tendency to select the more effective generalization strategy: opting for the greedy algorithm CART when CART performs better and the optimal algorithm GOSDT when GOSDT is the better choice. (c) We also conduct regression analysis showing that MetaTree's algorithmic preference positively correlates with the better generalizing algorithm's performance.

highlighting its capability to process categorical features and produce differentiable trees for LLM-generated inputs and user-queried outputs.

1.6 Analysis

After benchmarking the performance of MetaTree, we conduct an in-depth analysis of its behavior and splitting strategy. Our analysis of MetaTree begins by examining MetaTree's tendency between choosing a greedy split and an optimized split (Sec. 1.6.1). We measure the similarity of the generated splits between MetaTree and CART/GOSDT and evaluate all the splits' generalization performance. Turns out MetaTree would opt in for the better algorithm at the right circumstance. We then take a closer look at MetaTree's empirical bias-variance (Sec. 1.6.2). MetaTree illustrates a possibility to reduce empirical variance while not trading off empirical bias. We have included further analysis of evaluating MetaTree on synthetic XOR problems, considering the effects of noise and feature interactions (Sec. 1.6.5). And a look into MetaTree's internal decision-making process (Sec. 1.6.5) reveals MetaTree sometimes gets the right split early on, within the first 9 transformer layers.

1.6.1 Can MetaTree be less greedy when needed?

Our model is trained on mixed learning signals from GOSDT and CART, selected using generalization performance as the criterion. Our objective is to decipher whether MetaTree can strategically adapt its splitting approach between GOSDT and CART. To answer this question, we randomly take 100 samples (each of 256 data points) per dataset from the 91 left-out datasets and instruct MetaTree, GOSDT, and CART to generate splits for each sample. This approach ensures that all three algorithms are provided with the same data when making the splitting decision, allowing for a meaningful comparison.

We assess the similarity between the splits generated by MetaTree and those produced by GOSDT and CART. To measure it, we calculate the correlation coefficient between the label assignments following the splits. Additionally, we evaluate the generalization performance by examining the accuracy of the splits on its respective test set. We exclude samples where GOSDT and CART yield highly similar splits (correlation coefficient > 0.7).

We plot the output correlation between MetaTree and CART/GOSDT in Fig. 1.3a, for when CART is the better generalizing algorithm among the two. Similarly, we plot the output correlation between MetaTree and CART/GOSDT in Fig. 1.3b for when GOSDT is better generalizing. We divide the output correlation values into three categories (low, medium, and high correlation), and it can be observed that MetaTree tends to favor the algorithm that exhibits better generalization performance.

Furthermore, we visualize the relationship between the correlation difference (CART correlation minus GOSDT correlation) and the generalization performance difference (CART's test accuracy minus GOSDT's test accuracy) in Fig. 1.3c, noting that we exclude samples with marginal performance differences (generalization accuracy difference ≤ 0.08). A medium correlation (Pearson correlation = 0.64, p-value = 0.00003) is observed, suggesting a tendency in MetaTree's splitting strategy to align with generalization performance.

To better explain why MetaTree can learn to be non-greedy, at a high level, the model

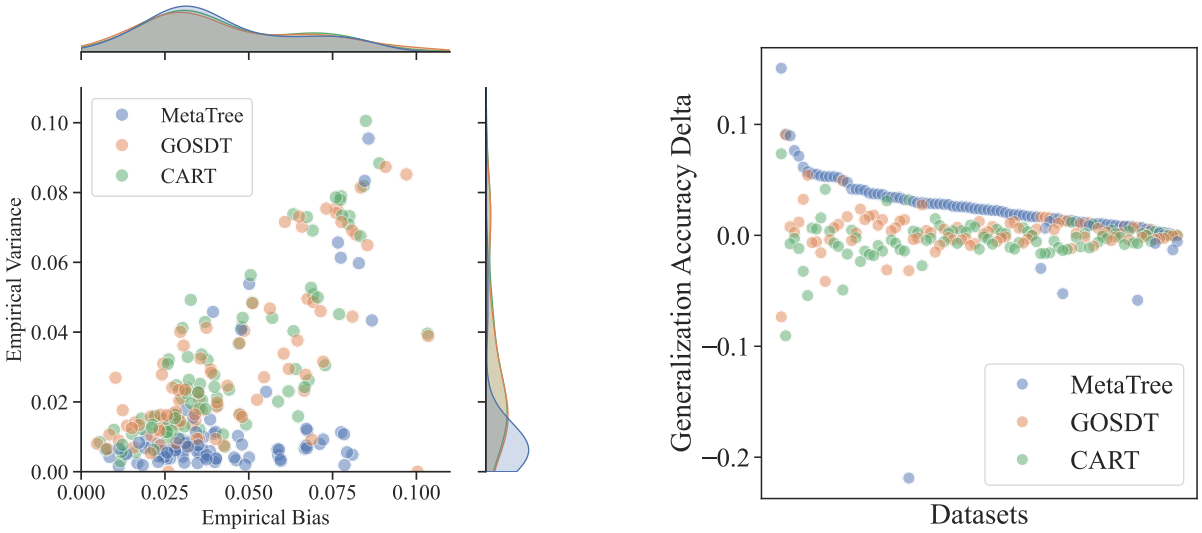
makes one decision at each node: shall I make the greedy split now? or is it better to plan for a split one step later? Due to our training strategy, MetaTree is optimized towards making these choices.

1.6.2 Breaking through the bias-variance frontier

Finally, we conduct a comprehensive bias-variance analysis to evaluate MetaTree, along with GOSDT and CART, on the 91 left-out datasets. This analysis shows how each algorithm navigates the trade-off between bias (the error due to insufficient learning power of the algorithm or incorrect model assumptions) and variance (the error due to sensitivity to small fluctuations in the training set). We use the empirical bias and variance as the measures in our evaluation; we perform 100 repetitions (N=100) for each dataset, therefore we have 100 decision tree models per algorithm per dataset. The empirical bias of an algorithm is calculated as the ℓ_2 difference between its produced models' average output and the ground truth labels, whereas the empirical variance is calculated as the mean ℓ_2 difference between its produced models' average output and each model's output.

The result is presented in Fig. 1.4a. Each point in the plot corresponds to the bias-variance coordinates derived from the performance of one algorithm on a single dataset. The x-axis represents empirical bias, indicating the algorithm's average error from the true function, while the y-axis corresponds to empirical variance, reflecting the sensitivity of the algorithm to different training sets. It can be observed that MetaTree demonstrates lower empirical variance, suggesting its robustness in diverse data distribution scenarios.

We accompany the bias-variance analysis with a dataset-level performance comparison for single trees of depth 2 in Fig. 1.4b. Each vertical axis represents a dataset. We first take the average generalization accuracy of CART and GOSDT as the base accuracy, then calculate the delta difference between each algorithm's accuracy and the base accuracy. Fig. 1.4b further illustrates MetaTree's advantage when comparing against GOSDT and CART.



(a) Empirical bias-variance analysis.

(b) Dataset level comparisons for single trees.

Figure 1.4. Empirical bias-variance decomposition for MetaTree, GOSDT, and CART on 91 left-out datasets with 100 repetitions is shown in (a). MetaTree has significantly lower variance and slightly smaller bias as compared to GOSDT and CART. (b) We compare accuracy delta (y-axis) for each dataset (x-axis) when fitting a single tree. The delta is the change compared to the mean accuracy of CART and GOSDT.

1.6.3 Memory Usage and Runtime Analysis

We conduct an analysis to show the memory usage and runtime difference among CART — an efficient algorithm with greedy heuristics and GOSDT — a mathematically guaranteed optimized algorithm and MetaTree— an transformer-based algorithm that learns from both worlds.

MetaTree enjoys the benefit of a non-recursive greedy algorithm — constant memory usage and fast inference speed, while having the superior performance among them all.

1.6.4 Ablation studies

We conduct ablation studies on our learning curriculum (two-phase v.s. single-phase, Sec. 1.6.5), learning signals (noisy real-world labels v.s. relabelled perfect synthetic labels, Sec. 1.6.5), training methodology (supervised training v.s. reinforcement learning, Sec. 1.6.5), positional bias (with positional bias v.s. without, Sec. 1.6.5) and our choice of

Table 1.2. We show the memory usage of CART, GOSDT and MetaTree when fitting decision trees with depth 2 & 3, using the memory profiler tool [Pedregosa and Gervais, 2021]. GOSDT takes a significant amount of memory since it is solving for the full decision tree solution on the data. *We report successfully terminated runs' stats, the experiments are conducted on a workstation with 125G memories and 128 cores.

		CART	GOSDT	MetaTree (GPU)
Depth=2	Mean Memory	0.82 MB	146.82 MB*	2420.48 MB
	Max Memory	1.41 MB	228.98 MB*	2464.87 MB
	Fail Rate	0%	1.1%	0%
Depth=3	Mean Memory	0.79 MB	15,615.69 MB*	2423.96 MB
	Max Memory	1.34 MB	87,721.02 MB*	2466.87 MB
	Fail Rate	0%	86.81%	0%

Table 1.3. We show the inference wall time for CART, GOSDT and MetaTree when fitting decision trees with depth 2. GOSDT takes a significant amount of time since it is solving for the full decision tree solution on the data. *We report successfully terminated runs' stats only.

		CART	GOSDT	MetaTree (GPU)
Depth=2	Mean Time	0.047s	26.44s*	0.090s
	Max Time	0.15s	74.42s*	0.102s

Gaussian smoothing hyper-parameter in Sec. 1.6.5. The summary of our ablation studies is put into Table 1.4 and the details for each ablation experiment can be found in the corresponding section.

1.6.5 Additional Experiments

Controlled setting: noisy XOR

We evaluate the MetaTree model in a controlled setting, on XOR datasets with level=$\{1,2\}$, label flipping noise=$\{0\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$, and dataset size 10k for each XOR level/noise ratio configuration. To assess performance, we employ the *relative error* metric, defined as the gap between the achieved accuracy and the maximum possible accuracy achievable ($= 100\% -$ label noise rate). Note that we have only trained our model on 10k trees generated with XOR Level 1 and 15% label noise. This specific training scenario was chosen to

Table 1.4. We conducted ablation studies on various important design choices, including training curriculum (two-phase versus single-phase, Sec. 1.6.5), learning signals (noisy real-world labels versus relabelled perfect synthetic labels, Sec. 1.6.5), training methodology (supervised training versus reinforcement learning, Sec. 1.6.5), positional bias (with positional bias versus without, Sec. 1.6.5) and our choice of Gaussian smoothing hyper-parameter. We summarize the mean and std of the test accuracy changes for evaluation on the 91 left-out datasets with 100 trees.

Ablation Setting	**Performance Change**
Curriculum: two-phase $\rightarrow$ single-phase	test acc $\downarrow$ -0.79 ± 0.14
Labels: noisy real-world $\rightarrow$ perfect synthetic	test acc $\downarrow$ -2.78 ± 0.23
Training: supervised $\rightarrow$ RL	training destabilized
Positional bias: with $\rightarrow$ without	test acc $\downarrow$ -0.8736 ± 0.04
Larger Gaussian radius: $\sigma = 0.05 \rightarrow \sigma = 0.1$	test acc $\downarrow$ -0.19 ± 0.06
Smaller Gaussian radius: $\sigma = 0.05 \rightarrow \sigma = 0.01$	test acc $\downarrow$ -0.33 ± 0.10

evaluate the model's robustness towards noise and adaptability to harder problems.

As indicated in Table 1.5b, MetaTree demonstrates a remarkable capacity for noise resistance and generalization. Once MetaTree learns to solve the XOR Level 1 problem, it can withstand much stronger data noise (while MetaTree has only seen 15% noise), and generalize to significantly harder XOR Level 2 problems (while MetaTree has only trained on XOR Level 1).

We further conduct a qualitative analysis, asking CART and MetaTree to generate decision trees for XOR Level 1&2 problems. The resulting trees, as depicted in Figure 1.5, offer insightful comparisons into the decision-making processes of both models under varying complexity levels.

Probing MetaTree's decision-making process

Our study is partially inspired by the logit-lens behavior analysis on GPT-2 [Hanna et al., 2023]. Their findings suggest that GPT-2 often forms an initial guess about the next token in its middle layers, with subsequent layers refining this guess for the final generation distribution. Building on this concept, we aim to investigate whether a similar guessing-refining pattern exists

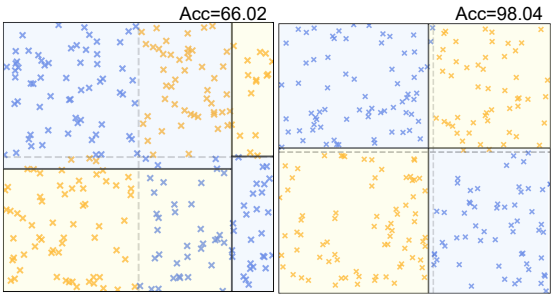
in our model and explore the feasibility of implementing an early exiting strategy as outlined in Zhou et al. [2020].

To this end, we analyze the decision-making process of our model at each Transformer layer. We can scrutinize how the model's decisions evolve by feeding the intermediate representation from each layer into the output module. One qualitative example is shown in Fig. 1.5c, where we ask MetaTree to generate the root node split on an XOR Level 1 problem. We can observe that the model gets a reasonable split right after the first layer. At layer 9, the model reaches the split that is close to its final output, and layer 10's output is an alternative revision with a close to ground truth split (the ground truth can be a vertical or horizontal split).

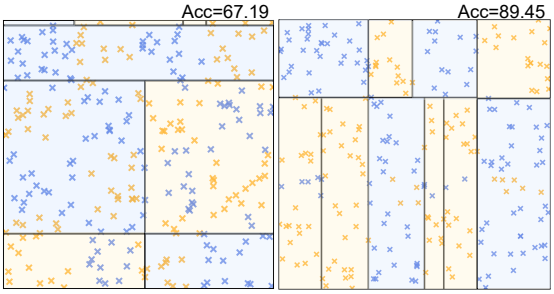
We proceed with a quantitative analysis to investigate the correlation between the model's final split and the splits occurring in its intermediate layers. We take samples from the 91 left-out datasets following the same procedure as detailed in Sec. 1.6.1, and we ask MetaTree to generate splits on them. The correlation between splits is determined by calculating the correlation coefficient between the label assignments after applying the splits. This metric essentially measures how closely aligned the splits are in dissecting the input regions.

The results of our quantitative analysis are presented in Fig. 1.5d. Notably, the correlation shows a gradual increase from layer 1 to 8, nearly reaching 1 at layer 9. However, it then drops significantly to approximately 0.2 at layers 10 and 11. This pattern suggests that the model consistently improves its ability to make accurate predictions in the initial 1 to 9 layers, while layers 10 and 11 may introduce some divergence or revision in its decision-making process.

This finding provides valuable insights into the internal decision-making process of MetaTree. It raises the possibility of considering early exit strategies at intermediate layers, particularly around layer 9, to enhance overall efficiency.



(a) Greedy Algorithm fails on XOR Level 1 (left), whereas MetaTree solves it (right).

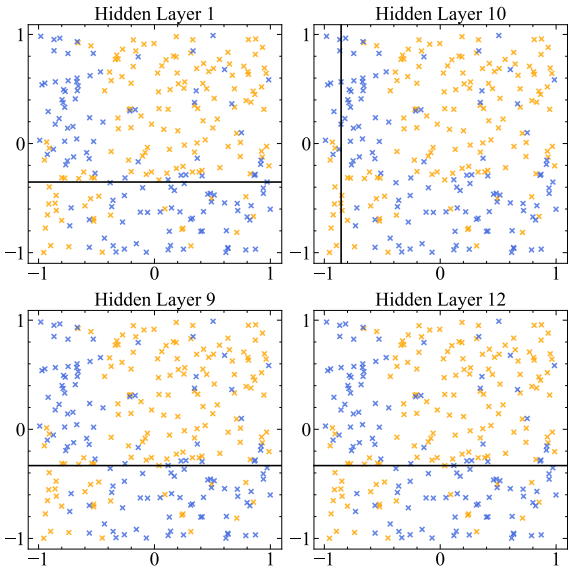


(b) Greedy Algorithm fails on XOR Level 2 (left), while MetaTree generalizes to it (right).

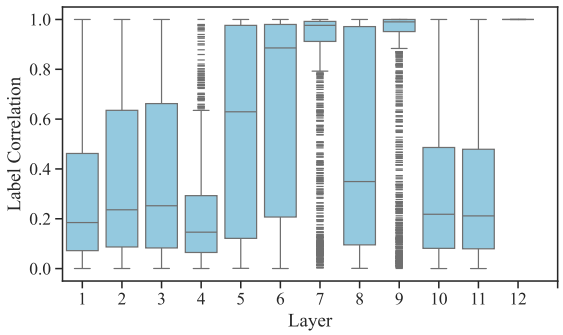
(a) Greedy algorithms such as CART cannot solve problems that require planning, such as Level 1&2 XOR. We show MetaTree can learn to solve Level 1 XOR and even generalize to solving Level 2 XOR.

Noise	0%	5%	10%	15%	20%	25%
XOR L1	3.59	3.26	2.94	2.64	2.37	2.04
XOR L2	12.78	11.06	9.34	7.32	4.99	2.05

(b) Relative error of MetaTree (trained on XOR Level 1 with 15% noise) on XOR datasets with level=$\{1,2\}$ and label flipping noise rate from 0% to 25%.



(c) Logit-lens probing of MetaTree on an XOR Level 1 problem.



(d) The output correlation analysis between the final split and each layer's splits of MetaTree across 91 left-out datasets.

Figure 1.5. Exploratory analysis of MetaTree. (a), (b) We examined MetaTree's performance in a controlled XOR setting with various noise levels and problem difficulty. We show an illustration of MetaTree solving Level 1 XOR and generalizing to Level 2 XOR while greedy algorithms like CART are unable to solve these. (c), (d) We probe the decision-making process of MetaTree over the Transformer layers, we found out that MetaTree can very often generate the final split early on.

Statistical Comparisons for Single-trees

To rigorously validate MetaTree's performance advantage beyond average rank comparisons, we conduct formal statistical hypothesis testing using the Fredman-Holm test [Demšar, 2006]. This non-parametric procedure is well-suited for comparing multiple algorithms across multiple datasets, as it accounts for the family-wise error rate when performing pairwise comparisons.

We test the null hypothesis that MetaTree and each baseline algorithm perform equally well in terms of single-tree generalization accuracy, across both the 91 held-out datasets and the 13 Tree-of-prompts datasets, at tree depths 2, 3, and 4. Results are shown in Table 1.5. The p-values are all below the significance threshold of 0.05, confirming that MetaTree's performance advantage over GOSDT, CART, ID3, and C4.5 is statistically significant. We note that GOSDT encounters out-of-memory errors at depths 3 and 4, preventing its inclusion in those comparisons.

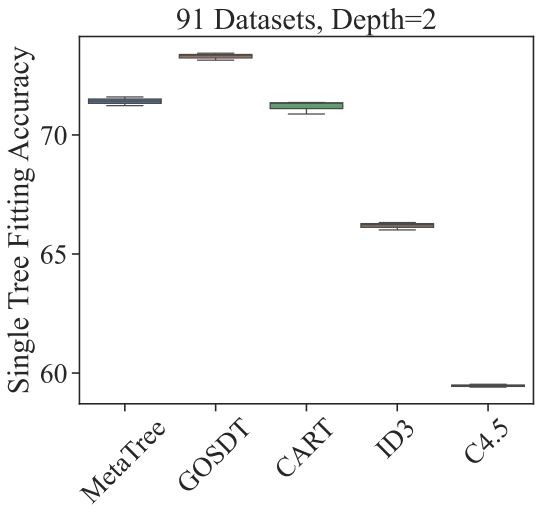
Single-tree fitting accuracy

While generalization accuracy measures how well the generated trees perform on unseen test data, fitting accuracy measures how well the trees fit the training data itself. A high fitting accuracy indicates that the algorithm can closely capture the patterns present in the training set, which is a necessary (though not sufficient) condition for strong generalization.

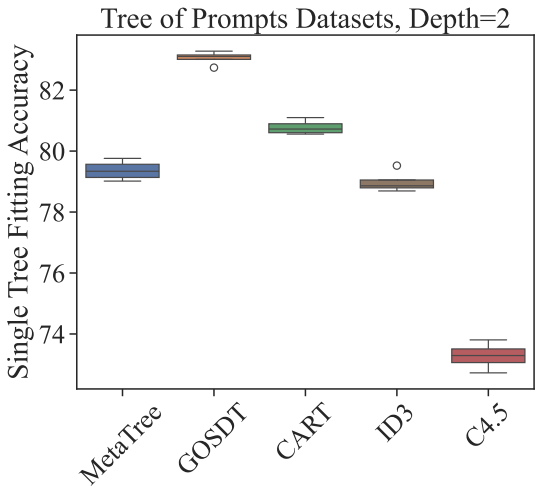
We show the fitting accuracy of the algorithms in Fig. 1.6 when generating single depth-2 decision trees over the datasets. On the 91 held-out datasets, MetaTree achieves the highest mean fitting accuracy, followed by GOSDT, CART, ID3, and C4.5. A similar trend is observed on the 13 Tree-of-prompts datasets. These results indicate that MetaTree is able to effectively utilize the training data to produce well-fitting trees, which, combined with its strong generalization performance shown in Sec. 1.5, suggests that MetaTree learns to fit the data without overfitting.

Table 1.5. Single-tree statistical comparison over the datasets with Fredman-Holm test Demšar [2006]. We are showing the p-values of the ranked tests. They are all of statistical significance (i.e. ≤ 0.05), demonstrating MetaTree's advantage over the baseline algorithms.

MetaTree v.s.	MetaTree	GOSDT	CART	ID3	C4.5
91D, depth=2	-	2.71e-08	8.88e-16	8.88e-16	0.00
ToP, depth=2	-	0.025	0.017	0.022	0.004
91D, depth=3	-	OOM	9.42e-12	4.19e-11	0.00
ToP, depth=3	-	OOM	0.014	0.011	0.028
91D, depth=4	-	OOM	2.86e-14	3.13e-10	0.00
ToP, depth=4	-	OOM	0.006	0.006	0.006



(a) Mean fitting accuracy over the 91 left-out datasets.



(b) Mean fitting accuracy over the 13 Tree-of-prompts datasets.

Figure 1.6. Average fitting accuracy for algorithms when generating single depth-2 trees.

Ablation Study: Training Curriculum

We devise a two-phase learning curriculum for MetaTree's training, as described in Sec. 1.3.2. In the first phase, the model is trained exclusively on GOSDT trees, allowing it to learn the behavior of the optimization-based algorithm. In the second phase, the training data is expanded to include both GOSDT and CART trees, selected based on which tree achieves better generalization on the validation set. The rationale behind this curriculum is that learning

from the more structured GOSDT trees first provides a strong foundation, and the subsequent exposure to both algorithms enables the model to learn when each approach is more effective.

Here we conduct an ablation study comparing our two-phase curriculum with a single-phase curriculum that trains directly on the GOSDT+CART dataset from the start, with all other configurations kept the same. The evaluation process is the same as the one used in Sec. 1.5: we compute their average test accuracy on 91 left-out datasets and for the number of trees from {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.

As shown in Fig. 1.7, the learning curriculum has a significant impact on the model at the end of training. The two-phase curriculum consistently outperforms the single-phase baseline across all ensemble sizes, with a particularly notable gap at smaller ensemble sizes. This suggests that the phased curriculum helps the model develop a stronger initial understanding of decision tree construction before learning to select between different algorithmic strategies.

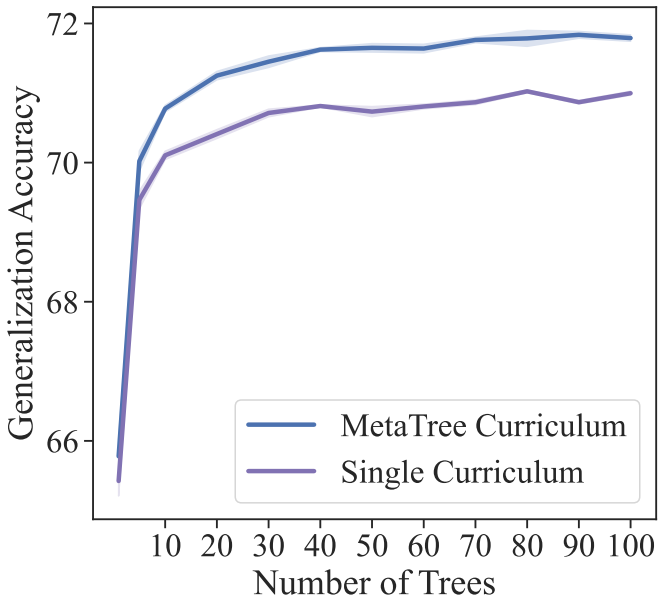


Figure 1.7. Average generalization accuracy for two different learning curricula. MetaTree's learning curriculum involves a first-phase training on GOSDT trees only, and a second-phase training on GOSDT+CART trees. The other curriculum is trained directly from GOSDT+CART datasets.

Ablation Study: Training Signal

In standard MetaTree training, the model learns from the original dataset labels, which may contain noise. An alternative approach is to train with noise-free data by relabeling each dataset with the predictions of the generated ground truth trees. By relabeling the dataset in this way, we can guarantee that the optimal decision splits are being fed into the model's training procedure, since every label perfectly aligns with the tree structure.

However, this relabeling removes the natural noise present in real-world data. While cleaner labels may simplify the learning problem, they may also reduce the model's ability to handle noisy inputs at test time. We evaluate a model trained with these synthetic labels on the 91 left-out datasets, for the number of trees from $\{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, and compare it against the standard MetaTree trained on noisy ground truth labels.

We show the results in Fig. 1.8. Learning from the original noisy labels brings better generalization capacity on unseen datasets. The model trained on synthetic labels shows a consistent performance gap, indicating that exposure to real-world label noise during training helps MetaTree build robustness and generalize more effectively to new datasets.

Ablation Study: Reinforcement Learning

In the early phase of our study, we experimented with training a model directly using reinforcement learning (specifically with proximal policy optimization), where the model can explore the space of decision tree splits and receive rewards as a function of the final generalization accuracy. This formulation is natural for decision tree construction: at each node, the model selects a splitting feature and threshold, and the quality of the complete tree can be evaluated on a held-out validation set to compute the reward signal.

However, we found that the training process is unstable due to the sparse reward and large search space. The reward is only available after the entire tree is constructed, making credit assignment to individual split decisions difficult. Furthermore, the combinatorial nature of the split space (choosing among all possible feature-threshold pairs at each node) makes exploration

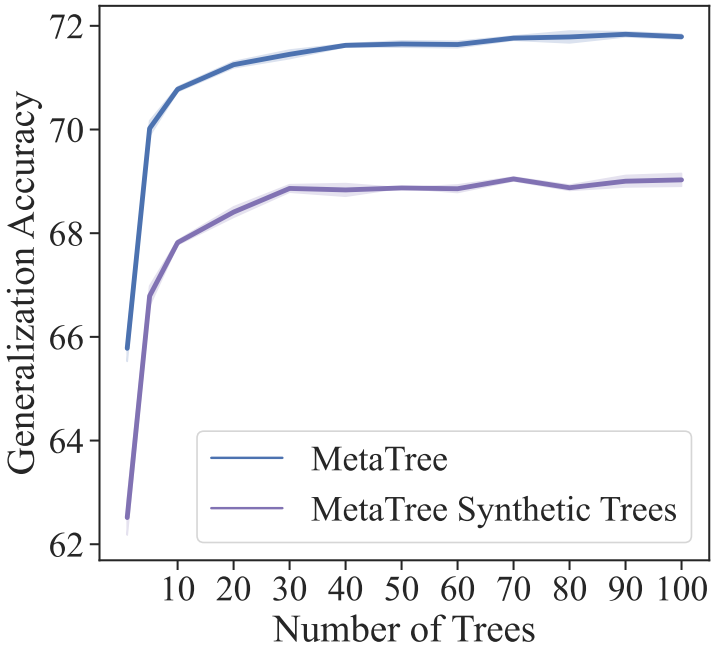


Figure 1.8. Average generalization accuracy for training with different labels. MetaTree trains directly from the noisy ground truth label. MetaTree Synthetic Trees is trained from the synthetic labels derived from the ground truth decision trees.

challenging. We plot 30 independent runs of such RL training in Fig. 1.9, where the task is to classify XOR L1 with only 2 dimensions and without any noise. Even on this simple problem, the RL training exhibits high variance across runs, with many runs failing to converge.

As can be seen, the training process is highly unstable and rarely reaches above 95% eval accuracy, whereas the supervised learning approach could quickly get to 97–98% eval accuracy on the same task. The instability is particularly concerning given that this is one of the simplest possible tree construction problems. For more complex datasets with higher-dimensional feature spaces, we expect these challenges to be even more pronounced. Therefore, we leave reinforcement learning for MetaTree as future research, noting that advances in reward shaping or hierarchical RL may help address these challenges.

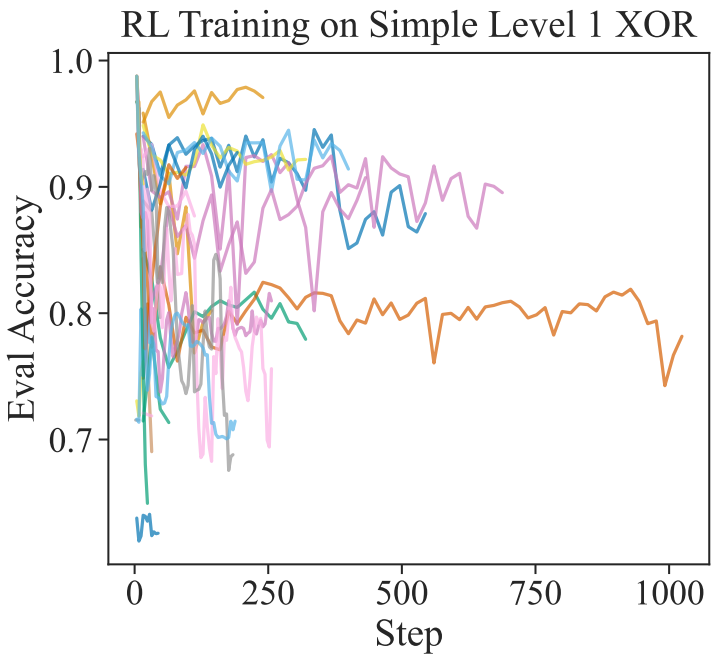


Figure 1.9. 30 runs of model training using reinforcement learning. It can be observed that the process is highly unstable.

Ablation Study: Positional Bias

As introduced in Sec. 1.3, MetaTree uses two positional bias b_1, b_2 at the input layer to anchor the row and column positions. We apply sequential and dimensional shuffling during training to make the model learn the positional invariance.

Alternatively, we can have a design that provides zero positional information to the model and becomes naturally invariant to input permutations. However, this design may have difficulties identifying the split, as the information will be mixed altogether over the Transformer layers.

We train a model with such a design, keeping all other training configurations the same, and compare its performance on the 91 left-out datasets using their average test accuracy for the number of trees from $\{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

As shown in Fig. 1.10, having such positional bias is empirically beneficial compared to a design that provides no positional information to the model. The model with positional bias consistently achieves higher generalization accuracy across all ensemble sizes. This performance

gap suggests that the learnable positional information helps the model better distinguish between different features and data points during the split decision process. Without any positional signal, the Transformer layers must rely solely on the content of the embeddings to determine which feature and threshold to split on, making it harder to precisely identify the optimal split location in the input table. The shuffling-based regularization used during training is sufficient to prevent the model from overfitting to specific positional patterns, while still allowing it to leverage positional cues for more accurate split decisions.

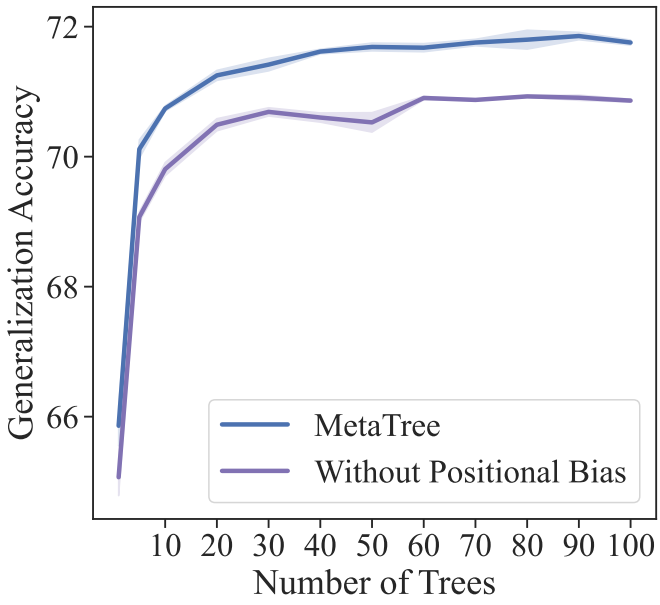


Figure 1.10. Average generalization accuracy for models trained with or without positional bias.

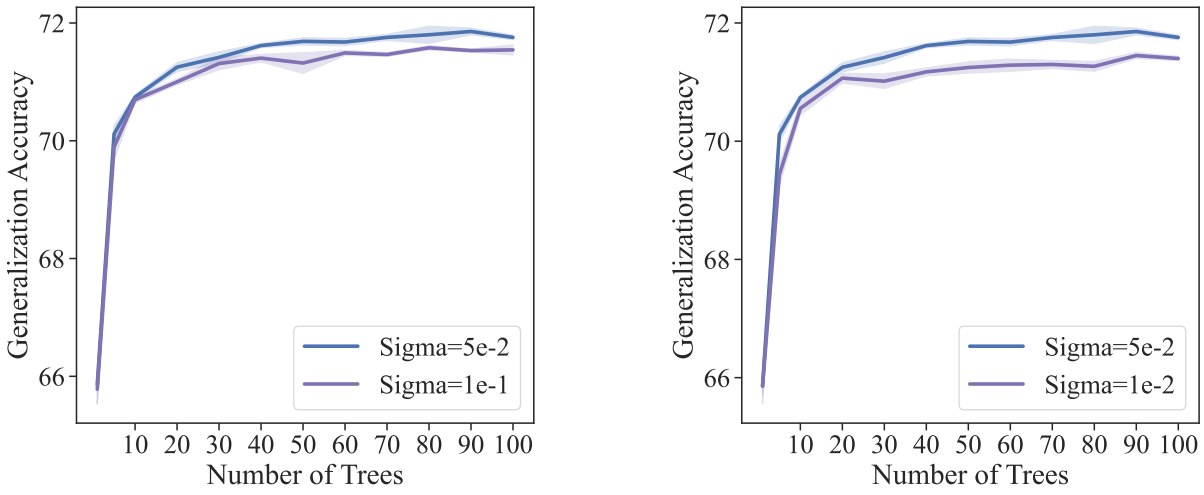
Ablation Study: Gaussian Smoothing Loss

We conduct an ablation study on the radius of the Gaussian smoothing loss (σ). It controls how much noise and signal the model receives during training. When σ is too large, every split may seem like a target split, but when σ is too small, the model may fail to learn from some equally good splits.

Following the evaluation pipeline as described in Sec. 1.5, we conduct our ablation study comparing our model trained using σ=5e-1 with models trained with a larger σ=1e-1 and a

smaller σ=1e-2, on 91 left-out datasets and compute their average test accuracy for number of trees from {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.

As shown in Fig. 1.11, the smoothing radius does have an impact on the trained model, and choosing an appropriate radius is essential to training an effective model. With a larger σ=1e-1, the smoothing distribution becomes too broad, causing the model to receive positive training signal for splits that are far from optimal. This makes it harder for the model to learn precise split boundaries. Conversely, with a smaller σ=1e-2, the smoothing is too narrow, and the model may fail to learn from splits that are equally good but differ slightly in their threshold value. Our chosen value of σ=5e-2 provides a balance between these two extremes, giving the model enough signal to learn from near-optimal splits while still maintaining precision in identifying the best split locations.



(a) Training with a larger σ=1e-1.

(b) Training with a smaller σ=1e-2.

Figure 1.11. Average generalization accuracy for models trained with different Gaussian Smoothing radius.

1.7 Discussion

Limitations and future directions

MetaTree is constrained by the inherent architectural limitations of transformers. Specifically, the maximum number of data points and features that MetaTree can process is bounded

by the transformer model's max sequence length (refer to Table A.1 for detailed specifications). However, these constraints can be alleviated by training a larger model. transformers' capacity for handling long sequences is constantly improving, with state-of-the-art LLMs [Reid et al., 2024] now able to take in sequences with up to 10M tokens. While MetaTree remains limited to small datasets, this work shows an important first step in learning to adaptively produce machine-learning models, and we leave training a large-scale LLM for future work.

Conclusion

We introduce MetaTree, a novel transformer-based decision tree algorithm. It diverges from traditional heuristic-based or optimization-based decision tree algorithms, leveraging the learning capabilities of transformers to generate strong decision tree models. MetaTree is trained using data from classical decision tree algorithms and exhibits a unique ability to adapt its strategy to the dataset context, thus achieving superior generalization performance. The model demonstrates its efficacy on unseen real-world datasets and can generalize to generate deeper trees. We conducted a thorough analysis of MetaTree's behavior and its bias-variance characteristics. More exploratory analysis and ablation studies are included in the appendix.

This work showcases the potential of deep learning models in algorithm generation, broadening their scope beyond predicting labels and into the realm of automated model creation. Its ability to learn from and improve upon established algorithms opens new avenues for research in the field of machine learning.

Acknowledgment

This work is sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints

for government purposes not withstanding any copyright annotation hereon.

Chapter 1, in full, is a reprint of the paper: Zhuang, Yufan, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. "Learning a Decision Tree Algorithm with Transformers." Transactions on Machine Learning Research. 2024. The dissertation author was the primary investigator and lead author of this paper.

Chapter 2

In-context Learning with Continuous Vector Representations

Large language models (LLMs) have shown remarkable in-context learning (ICL) capabilities on textual data. We explore whether these capabilities can be extended to continuous vectors from diverse domains, obtained from black-box pretrained encoders. By aligning input data with an LLM's embedding space through lightweight projectors, we observe that LLMs can effectively process and learn from these projected vectors, which we term Vector-ICL. In particular, we find that pretraining projectors with general language modeling objectives enables Vector-ICL, while task-specific finetuning further enhances performance. In our experiments across various tasks and modalities, including text reconstruction, numerical function regression, text classification, summarization, molecule captioning, time-series classification, graph classification, and fMRI decoding, Vector-ICL often surpasses both few-shot ICL and domain-specific model or tuning. We further conduct analyses and case studies, indicating the potential of LLMs to process vector representations beyond traditional token-based paradigms.[1]

2.1 Introduction

In-context learning (ICL) has emerged as a powerful paradigm in large language models (LLMs), allowing generalization from limited examples within a given context [Brown et al.,

[1] Code is available at: https://github.com/EvanZhuang/vector-icl.

2020, OpenAI, 2023]. By providing demonstrations in the context during inference, ICL allows models to adapt to new tasks and formats without the need for retraining. However, since LLMs are trained on discrete natural language tokens, ICL is generally learned and used through natural language, limiting its applicability to non-textual data.

We explore whether LLMs can perform ICL directly on continuous vectors, a capability that could dramatically expand their applicability. Many data modalities, such as sensor readings, financial time series, or scientific measurements, lack a natural text representation. Moreover, even for text data, information like numbers might be better represented via continuous vectors than tokens.

In our study, we observe that LLMs can indeed understand and process continuous context via embedding projection. This technique, which we term Vector-ICL, acts as a bridge between continuous data and the LLM's embedding space. Simple linear projections are often sufficient, though for cross-modal tasks—such as those involving non-textual data like time-series or graphs, non-linear transformations may be required. We demonstrate that training the embedding projector using a straightforward next-token prediction objective enables Vector-ICL, effectively teaching the LLM to "read" continuous vectors. Moreover, fine-tuning the projector on downstream tasks further enhances the effectiveness of continuous context, outperforming few-shot ICL and domain-specific models or tuning.

Our investigation begins with the task of text reconstruction, where we assess whether LLMs can recover information encoded in text embedding. This serves as a proof-of-concept for Vector-ICL, showing that LLMs can indeed extract meaningful information from projected continuous vectors. We then turn to the more complex challenge of arithmetics. Although state-of-the-art LLMs can solve Olympiad mathematical problems [Trinh et al., 2024, OpenAI, 2023], they struggle with precise number processing due to the limitations inherent in their tokenization schemes. Our results demonstrate that Vector-ICL offers a more effective approach for function approximation, particularly for large numbers that span multiple tokens, potentially opening new avenues for enhancing LLMs' numerical reasoning capabilities.

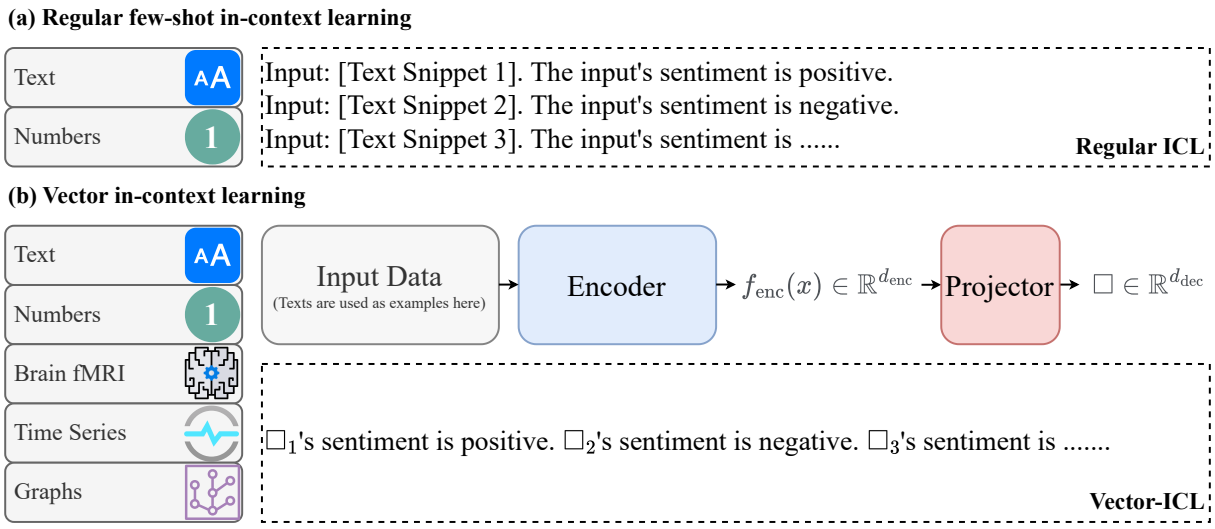


Figure 2.1. Comparing regular in-context learning to vector in-context learning. (a) In regular ICL, textual demonstrations are given as context during LLM inference. (b) In Vector-ICL, the input space is extended across multiple modalities. The input data is first encoded as embeddings, then transformed into continuous vectors which represent as box tokens (□) via embedding projection. During inference, we provide box tokens in prompts as demonstrations for ICL. We consider box tokens representing text, numerical data, brain fMRI, time series, and graphs in this study.

Finally, we extend our analysis to a broad range of modalities and tasks, including text classification, summarization, molecule captioning, brain fMRI reconstruction and classification, time-series classification, and graph classification. Across these diverse domains, LLMs exhibit competitive and often superior performance when employing Vector-ICL, revealing previously untapped capabilities of these models. This work highlights the potential of continuous representations in enhancing LLMs' in-context learning capacities, pushing the boundaries of what these models can achieve beyond token-based paradigms.

2.2 Related work

Empirical results of in-context learning

ICL has empirically shown strong performance in diverse natural-language processing tasks with very few demonstrations [Brown et al., 2020, OpenAI, 2023]. In modern LLMs with

long context windows, ICL has even shown performance improvements as the number of demonstrations grows to hundreds or even thousands, sometimes outperforming finetuning [Agarwal et al., 2024, Li et al., 2023b, Bertsch et al., 2024]. Empirically, different factors play key roles in ICL. In smaller LLMs, ground-truth demonstrations are not required for in-context learning, while other factors such as the label space, input text distribution, and overall sequence format play an important role [Min et al., 2022b]. Moreover, these LLMs can sometimes achieve strong performance even when demonstrations are intentionally irrelevant or even pathologically misleading [Webson and Pavlick, 2022]. Flan-T5 [Chung et al., 2024] revealed that instruction tuning improves few-shot learning by helping LLMs better utilize in-context examples in an encoder-decoder architecture.More recently, Wei et al. [2023] characterize these behaviors of LLMs with respect to model size, and show that larger language models perform in-context learning differently in the presence of flipped or semantically unrelated labels. Orthogonally, different works find ways to improve ICL, e.g. by including explanations [Lampinen et al., 2022], or chaining ICL calls [Morris et al., 2023]. ICL has shown some success in multimodal models [Wu et al., 2024, Jiang et al., 2024] or when applied to tabular data [Zhao et al., 2024b].

Understanding ICL

Many works have investigated ICL and found that it is able to learn linear models [Akyürek et al., 2022, Zhang et al., 2023], discrete functions [Bhattamishra et al., 2023], and more general algorithms [Li et al., 2023c]. Some works have explicitly connected ICL in specific settings to implementing optimization steps analogous to gradient descent [Mahankali et al., 2023, Von Oswald et al., 2023, Ahn et al., 2024] and higher-order optimization methods [Dai et al., 2023, Fu et al., 2023, Giannou et al., 2024, Zhang et al., 2023]. A complementary direction aims to establish statistical complexity and generalization bounds of in-context learning in transformers [Bai et al., 2024, Li et al., 2023c, Wies et al., 2024, Wu et al., 2023]. Finally, one recent work suggests that ICL may arise from parallel structures in pretraining data [Chen et al., 2024].

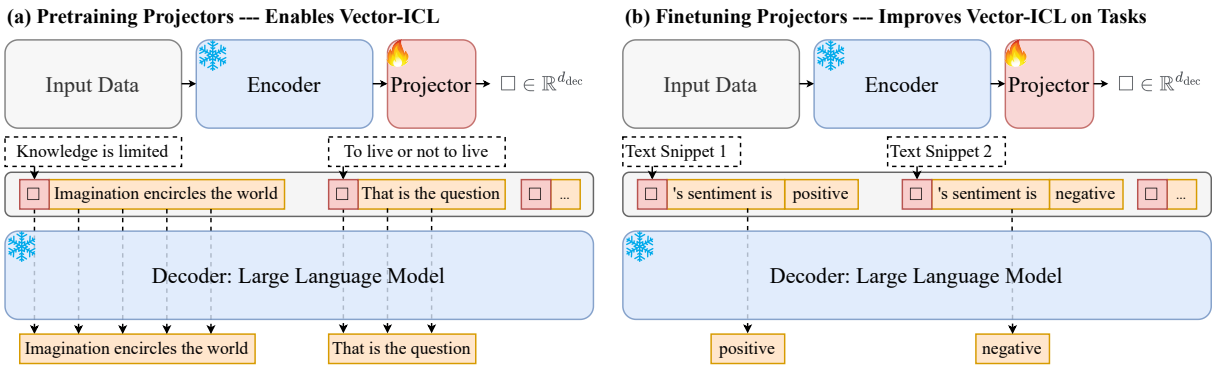


Figure 2.2. Pretraining and finetuning the projectors. Vector-ICL requires updating the parameters of a lightweight projector while keeping the encoder and decoder parameters fixed. The encoder first compresses the input into single token embeddings, and then the projector will project it to the aligned representation space for LLMs' later use. (a) Pretraining the projector on a general language modeling corpus (or a modality-to-text dataset) enables Vector-ICL. (b) Task-specific fine-tuning makes Vector-ICL outperform few-shot ICL on natural language tasks, as well as with domain-specific models on non-language tasks.

Learning to learn in-context

In contrast to the emergent ICL capabilities of LLMs, existing works have also studied how to explicitly improve ICL. Min et al. [2022a] propose MetaICL, a meta-training framework for finetuning pretrained LLMs to perform in-context learning on a large and diverse collection of tasks. In the tabular domain, TNP [Nguyen and Grover, 2022] and PFNs [Müller et al., 2021] train transformer models to perform in-context prediction for a family of functions, which allows in-context generalization to unseen functions after training. Zhao et al. [2023] also propose meta-learning transformers to in-context learn group preferences, serving as an in-context learned reward model that adapts to diverse group preferences.

2.3 Method: vector context via embedding projection

2.3.1 Embedding projection

Vector-ICL requires transforming inputs into vector contexts through an embedding projection. Given a dataset $\mathscr{X} = \{x_i\}_{i=1}^n$, we assume the existence of an encoder f_{enc}, that transforms the data into an abstract representation (alternatively, the raw data may already be

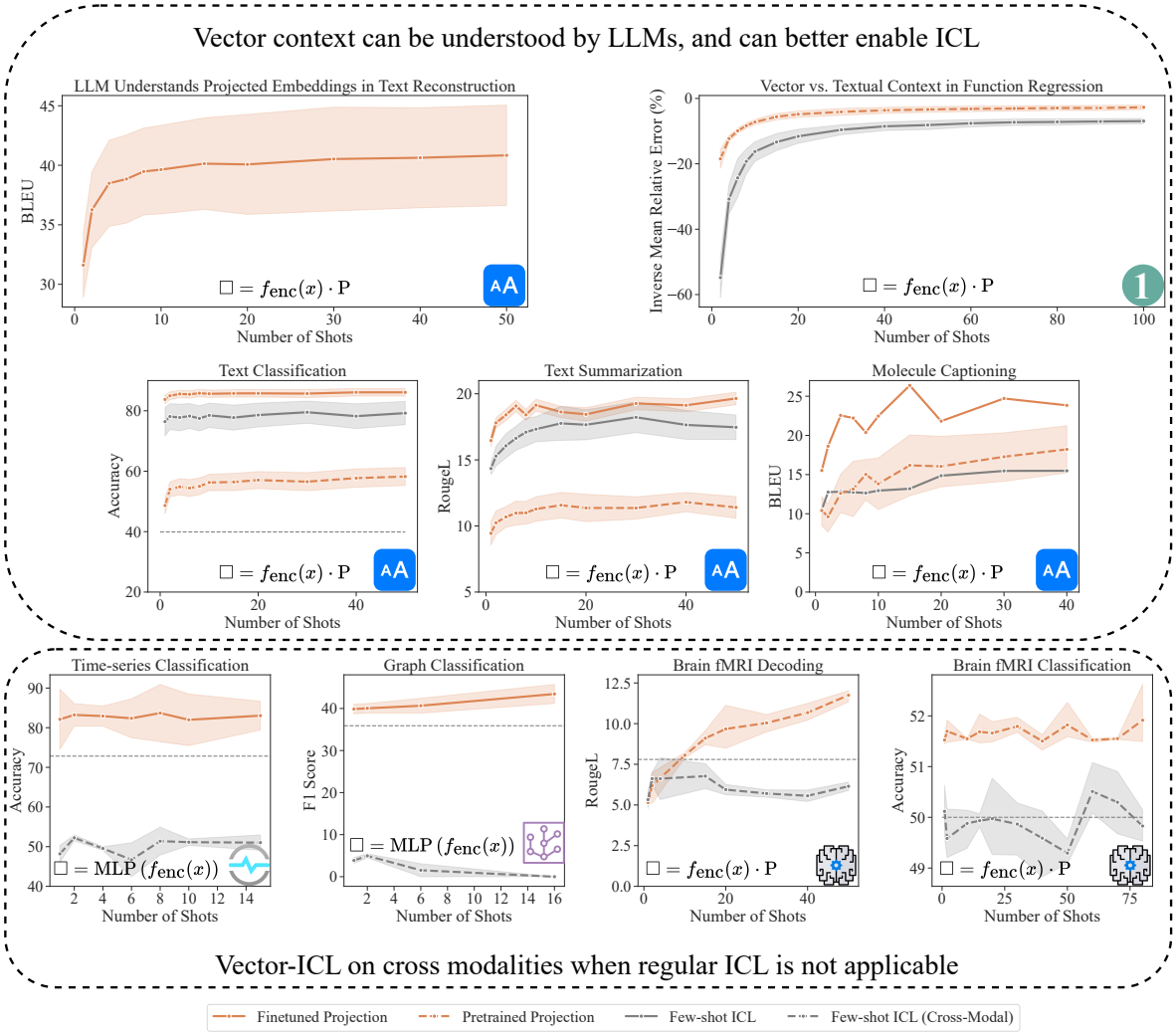


Figure 2.3. Main results: LLMs can perform Vector-ICL (↑ = better). We show that training the embedding projector with a simple next-token prediction objective enables Vector-ICL. Even with only unsupervised pretraining, Vector-ICL matches or outperforms traditional few-shot ICL on 4 out of 6 tasks where direct comparison is possible. Fine-tuning the projector on downstream tasks further enhances the use of continuous context, consistently surpassing both few-shot ICL and specialized task-tuned baselines (soft-prompt for text, tuned encoders for non-text).The study begins with text reconstruction to assess LLMs' ability to interpret box token embeddings, followed by function regression to evaluate reasoning capabilities. We then demonstrate Vector-ICL's effectiveness and applicability across various downstream tasks, including text classification, summarization, time-series classification, graph classification, and brain fMRI decoding & classification. Results in each panel are averaged over different encoders and LLMs for the diverse tasks we study; error bars show 95% confidence intervals.

a continuous vector). The encoded embeddings, $f_{\text{enc}}(x)$, are then projected into box tokens, denoted as $\square_x$. Throughout the paper, we will use the terms "box tokens" and "projected embeddings" interchangeably. For decoding, we use a language model f_{llm} that generates outputs based on the provided prompts.

We impose no constraints on the form of the input data $\mathscr{X}$; it can come from any modality. The only requirement is that the encoder f_{enc} maps each data point x into a vector space, defined as:

$$f_{\text{enc}} : x \to \mathbb{R}^{d_{\text{enc}}},\ \forall x \in \mathscr{X} \tag{2.1}$$

The LLM typically processes discrete tokens $\{\text{tok}_1, \text{tok}_2, \ldots, \text{tok}_l\}$, then maps them to text embedding space $\{\text{emb}_1, \text{emb}_2, \ldots, \text{emb}_l\}$, $\forall i$, $\text{emb}_i \in \mathbb{R}^{d_{\text{dec}}}$. Since we operate mostly in embedding space, we omit the tokenization step for simplicity and directly refer to text inputs as their embedding representations.

The process of embedding projection is then carried out as follows. For linear projection, we construct a projection matrix $\text{P} \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{dec}}}$ and make the following transformations to obtain projected embedding $\square_x$ given input x:

$$\square_x := f_{\text{enc}}(x) \cdot \text{P} \tag{2.2}$$

In cases where more expressive power is needed, we utilize a two-layer multi-layer perceptron (MLP) to perform the projection:

$$\square_x := \text{MLP}(f_{\text{enc}}(x)) \tag{2.3}$$

The MLP follows the architecture of the MLP block in Llama [Touvron et al., 2023b], with an additional input projection layer to map the input dimension $\mathbb{R}^{d_{\text{enc}}}$ to output dimension $\mathbb{R}^{d_{\text{dec}}}$.

2.3.2 Projected embeddings as context

The projected embeddings are then utilized as context in Vector-ICL, functioning as the equivalent of the original input data. For example, in NLP tasks, the original text snippets x will first be encoded as embeddings $f_{\text{enc}}(x)$, then projected to become $\square_x$, inserted into the prompt like the following:

$\square_x$'s sentiment is ... / $\square_x$'s summarization is ...

Using them as the context in ICL is then natural:

$\square_1$'s sentiment is happy. $\square_2$'s sentiment is sad. $\square_x$'s sentiment is ...

where $\square_1$ and $\square_2$ are in-context examples and $\square_x$ is the input.

2.3.3 Training the embedding projectors

The projectors need to be trained to achieve effective projections. We discovered that pretraining these projectors with language modeling objectives enables the ICL capabilities with vector context, and finetuning them on task datasets further improves ICL performance.

The pretraining process is depicted in Fig. 2.2(a). For each text snippet, we cut it into two pieces with the cutting point randomly sampled from the end of sentences. The first half is encoded and projected while the second is kept intact. The rest is the same with any pretraining process, the language model generated the next token distribution at each input position, except for the ones preceding the projected embeddings, and a cross-entropy loss is imposed on top of this. With the encoder and LLM frozen, the gradient backpropagates to the projector, updating its parameters.

For non-text data modalities, pretraining can be more flexible. We define this pretraining as involving general, non-task-specific objectives, such as reconstructing a number from its embeddings (e.g., $\square_x$ is $\underline{32768}$), performing basic algebra (e.g., $\square_x + \square_y = \underline{16384}$), or predicting the next token from brain fMRI embeddings.

The finetuning process is shown in Fig. 2.2(b). It utilizes additional structured prompts

and trains with task-specific datasets. Similarly, the input is first mapped into the embedding space and projected into □ tokens. They are then inserted into structured prompts, while the projector is trained with conditional generation loss given those prompts.

2.4 Experimental setup

Table 2.1 gives an overview of our experimental setup, including specifics for the task, datasets, encoders, LLMs, and task-specific prompts we use. Across different tasks, we project to four open-weights LLMs. We now provide details for individual tasks.

Baselines

We evaluate Vector-ICL in two distinct settings: with and without task-specific tuning. For textual tasks, we compare against few-shot ICL and soft prompt tuning [Li and Liang, 2021]. We choose soft prompt tuning as our primary baseline because it represents a similarly lightweight adaptation approach - both methods introduce a small number of trainable parameters while keeping the base LLM frozen. Like our projector, soft prompts modify how the LLM processes inputs without changing its internal weights. This makes it a fair comparison point for assessing whether Vector-ICL's benefits come from the continuous vector representations themselves rather than just the additional training. For non-textual tasks, where soft prompts cannot be directly applied, we compare against cross-modal few-shot ICL and tuned encoders. In the non-textual domain, ICL inputs are represented either as numeric sequences (for time-series and brain fMRI data) or as textual descriptions (for graph edge lists and node features).

Text Pretraining

To pretrain our text projectors, we leverage the WikiText-103 [Merity et al., 2016] dataset, consisting of over 100 million tokens from verified high-quality Wikipedia articles. This smaller language modeling corpus is chosen for its suitability to the lightweight nature of our projectors. The pretraining process is illustrated in Fig. 2.2(a), where text snippets are divided at random sentence-end points. The first half is embedded and projected, while a next-token generation

Table 2.1. Overview of the tasks, datasets, encoders, large language models, and prompts used in the experiments. Each task utilizes encoders and LLMs to perform functions across multiple modalities. The table highlights the diversity of models and configurations applied to each task.

Task	Dataset	Encoder	LLM	Prompt
Text Reconstruction	Quora [Thakur et al., 2021] PST [Tiedemann, 2012]	N, S, S, G		Translate the text in brackets: (□) Translation: [Text]
Function Regression	10 Digits Regression	Digit Embedding	, , ,	$x = \square_x$, $y = \square_y$, function(x, y) equals to (digits): [Solution]
Text Classification	IMDB [Maas et al., 2011] Rotten Tomatoes [Pang and Lee, 2005] SST2 [Socher et al., 2013] Emotion [Saravia et al., 2018] Financial Phrasebank [Malo et al., 2014]	N, S, S, G	, ,	(□)'s sentiment is: [Label]
Text Summarization	XSum [Narayan et al., 2018] XLSum [Hasan et al., 2021]	N, S, G		(□)'s summarization is: [Summary]
Molecule Captioning	Language + Molecule-24 [Edwards et al., 2024]	N		(□)'s molecule caption is: [Caption]
Brain fMRI	LeBel et al. 2022, Tang et al. 2023	PCA of fMRI		[Question] Input: □ Response: [Answer]
Time-series Classification	FordA, FordB [Dau et al., 2019]	Chronos-base [Ansari et al., 2024]		(□)'s class (positive, negative) is: [Label]
Graph Classification	ogbg-molhiv [Hu et al., 2020]	Graphormer [Ying et al., 2021]	,	(□)'s class (positive, negative) is: [Label]

Encoders: N NV-Embed (Lee et al. 2024; `nvidia/NV-Embed-v1`), S SFR (Meng et al. 2024; `Salesforce/SFR-Embedding-2_R`), S Stella (Zhang 2024; `dunzhang/stella_en_1.5B_v5`), G GTR-t5 (Ni et al. 2021; `sentence-transformers/gtr-t5-base`)
LLMs: Llama-3.1-8B (Dubey et al. 2024; `meta-llama/Llama-3.1-8B-Instruct`), Mistral-7B (Jiang et al. 2023; `mistralai/Mistral-7B-Instruct-v0.3`), Qwen2-7B (Yang et al. 2024; `Qwen/Qwen2-7B-Instruct`), Yi-1.5-9B (Young et al. 2024; `01-ai/Yi-1.5-9B-Chat`)

loss is applied to the second half.

Text Reconstruction

We investigate LLMs' ability to decode original text from projected embeddings using two datasets: Parallel Sentence Talks [Tiedemann, 2012] and Quora [Thakur et al., 2021]. These datasets consist of concise text pieces that convey clear meaning. Projectors are trained on the training sets of both datasets, and performance is evaluated using the BLEU score [Papineni et al., 2002, Post, 2018].

Arithmetic and Function Regression

For the arithmetic tasks, we generated synthetic datasets containing 10-digit numbers, which are particularly challenging for LLMs as they require splitting the numbers into multiple text tokens. These numbers are represented using a concatenated one-hot encoding per digit. For

instance, a 10-digit number is represented as a 10×10 matrix, flattened into a 100-dimensional vector. The pretraining phase includes two key tasks: number reconstruction, where the model is tasked with recovering the original number from its embedding, and basic arithmetic, where the model performs algebraic addition operations on the projected embeddings.

To evaluate the models' arithmetic reasoning abilities, we employ a non-linear function regression task, where the function is defined as $f(x,y) = \sqrt{x}\sqrt{y}$. The model is provided with inputs x and y, and it must predict the integer part of the function output. Performance is measured using the mean relative error, calculated as the ℓ_1 difference between the predicted and true values, normalized by the ground truth. This task allows us to assess the models' ability to perform more complex numerical reasoning beyond simple arithmetic operations.

Text Classification

We assess whether Vector-ICL can be applied effectively to text classification. Both binary and multi-class classification datasets are used, and the results are compared across few-shot ICL and soft prompt tuning. The classification performance is measured by accuracy.

Text Summarization

Following the classification tasks, we explore Vector-ICL's capability in summarizing text based on the projected embeddings. The datasets, encoders, LLMs, and prompt templates can be found in Table 2.1. Performance is evaluated using RougeL [Lin, 2004].

Molecule Captioning

We also extend our approach to the unconventional task of molecule captioning, using molecule sequence-caption pairs from the Language + Molecules-24 (LPM24) [Edwards et al., 2024] dataset. A sample molecule-caption pair looks like the following:

Molecule: Cc1c(Cl)cccc1-n1ccn2c(SCC(=O)c3ccccc3C(F)(F)F)nnc2c1=O

Caption: The molecule is a pain treatment that impacts inflammatory disease treatment.

This task explores whether LLMs can extract useful information from projected embeddings of out-of-distribution chemical sequences, with performance evaluated via BLEU

score.

Brain fMRI Decoding and Classification

We analyze data from LeBel et al. 2022 and Tang et al. 2023, which consists of fMRI responses for 3 human subjects as they listen to 20+ hours of narrative stories from podcasts. We preprocessed the data following Benara et al. 2024, by converting the fMRI responses into a 200-dimensional output using principal components analysis and assigning classification labels to 10-grams of the story text at 2-second intervals using an LLM.

We use the same pretraining methodology as text to pretrain on the brain fMRI data (projecting on 20% of time points and imposing next-token generation loss on the remaining 80%) . We evaluate the LLM's capability to decode projected brain fMRI by giving them randomly sampled context from the train set, that could come from different human subjects or from a different story, and ask them to decode segments from the test set.

In addition to text reconstruction, we decode the binary labels from the fMRI responses, corresponding to questions about the underlying text, e.g. "Does the sentence contain a proper noun?" The decoding random baseline is constructed by giving the LLM the randomly sampled, shuffled text from the training set, and generating text according to it. We measure the performance using the RougeL score between the generated text and the ground truth text. The classification random baseline is 50% accuracy, as we have balanced the dataset.

Time-series

We take the output of the last time step from Chronos-base [Ansari et al., 2024] as the time-series' representation. We use the base encoder with trained classification head as the baseline and we measure the prediction performance with accuracy.

Graphs

We use Graphormer [Ying et al., 2021] as the encoder model, specifically the one that was pretrained on quantum chemistry graph datasets [Hu et al., 2021]. Since the down-stream, ogbg-molhiv [Hu et al., 2020], is a molecule property prediction dataset, and with strong

Table 2.2. Comparison of Vector-ICL, few-shot ICL, and soft prompt tuning across various sentiment analysis and summarization datasets. Details can be found in Sec. B.1.

Method	Sentiment Analysis					Summarization	
	Rotten Tomatoes	SST2	IMDB	Emotion	Financial Phrasebank	XSum	XLSum
V-ICL (pretrained)	80.60	78.90	95.04	41.20	60.72	15.25	15.89
Few-shot ICL	87.31	91.74	93.50	55.20	71.78	19.53	19.41
Soft Prompt	**93.24**	96.21	95.26	74.15	78.22	12.84	12.70
V-ICL (finetuned)	88.80	**98.16**	**97.28**	**85.20**	**81.68**	**20.08**	**20.49**

class imbalance (3-4% positive classes), we finetune the encoder on the training set to provide meaningful baselines and embeddings. We take the output prior to the classification layer of the Graphormer as the graph embedding. Weighted sampling is adopted in the finetuning of both the baseline Graphormer and the embedding projector to yield meaningful predictions. We use the finetuned Graphormer as our baseline and use the F1 score as the performance metric.

Projector Configurations

Both linear and non-linear projectors are utilized, as shown in Fig. 2.3, with input and output dimensions matching the encoder-decoder pairs. Early stopping with patience of 500 steps is used during finetuning, as projectors converge quickly due to their small parameter sizes. Details of the hyperparameters used in training are provided in Table B.1.

2.5 Results: unlocking versatile applications across modalities

Fig. 2.3 presents our main results, where each subplot corresponds to one of the nine tasks. We begin by exploring text reconstruction, to see whether LLMs can comprehend the information encoded within the box tokens. Next, we investigate the tasks of function regression to evaluate whether LLMs can leverage the box tokens during reasoning processes and whether this approach outperforms reasoning with plain text. Finally, we proceed to a range of downstream tasks, including text classification, text summarization, time-series classification, graph classification,

and brain fMRI decoding. This comprehensive evaluation allows us to assess the versatility and effectiveness of Vector-ICL across different domains and task types.

Text Reconstruction: LLM Understanding of Projected Embeddings

We first verify LLMs' ability to understand projected embeddings. Our results demonstrate that Vector-ICL successfully reconstructs original text from projected embeddings, with performance improving as the number of examples (shots) increases, mirroring standard in-context learning (ICL) behavior. This suggests that LLMs can effectively decode the information compressed into the box tokens, with more context leading to better reconstruction. The similarity to ICL behavior indicates that Vector-ICL leverages similar learning mechanisms, but with the flexibility of working with continuous representations.

Function Regression: Enhanced Reasoning with Continuous Context

We then study would it sometimes be better for LLMs to receive continuous context instead of discrete tokens. After pretraining projectors on 10-digit number reconstruction and addition between two numbers, we task the LLM with learning an unknown function in-context. Results show Vector-ICL consistently outperforms few-shot ICL with raw number inputs, that have to span multiple tokens. This suggests that the continuous representations capture numerical relationships more effectively than discrete tokens, enabling LLMs to better infer and apply mathematical patterns. The improvement is particularly noteworthy given that LLMs are typically challenged by precise numerical computations.

Text Classification

In this classical NLP task, we aggregate mean accuracy across five datasets, four encoders, and three LLMs. Results indicate that pretrained projectors provide meaningful continuous context for ICL, outperforming the random baseline. LLMs achieve optimal performance with continuous context from finetuned projectors, surpassing both regular few-shot ICL and soft prompt tuning, with details shown in Table 2.2. This demonstrates the versatility of Vector-ICL across different text classification scenarios and its ability to outperform established prompt

tuning methods. The success across multiple datasets and LLMs suggests that the benefits of continuous context are robust and generalizable.

Text Summarization

This task demands longer text generation and deeper information comprehension. We aggregate mean RougeL scores across two datasets, and three encoders. Findings show that pretrained projectors enable LLMs to extract and condense information from a single box token, while finetuned projectors provide more effective continuous context than original textual input and soft prompts, with details shown in Table 2.2. The ability to compress and later expand information from a single token is particularly impressive, suggesting that Vector-ICL captures high-level semantic content effectively. The superior performance of finetuned projectors highlights the benefits of task-specific optimization in continuous space.

Molecule Captioning: An Unconventional NLP Task

We explore LLMs' ability to comprehend continuous vector context for out-of-distribution inputs like molecule sequences. Evaluating captioning performance with BLEU scores, we find that both pretrained and finetuned projectors provide better context than original molecule sequence text, despite encoders likely never trained on such sequences. This result is particularly intriguing as it demonstrates Vector-ICL's ability to bridge the gap between specialized domains and general language understanding. It suggests that continuous representations can capture and translate domain-specific information in a way that's more accessible to LLMs than raw specialized notation.

Time-series Classification

We finetune non-linear projectors on the training sets of two datasets and evaluate LLM performance with the resulting continuous context. Aggregating average accuracy, we find LLMs outperform baseline domain-specific models with finetuned classification heads. The success here suggests that continuous context can effectively capture temporal dependencies and patterns, translating time-series data into a form that LLMs can process effectively.

Graph Classification

Since the task dataset is heavily imbalanced and out of the pretraining distribution of the graph encoder. We first finetune the base encoder on the target dataset to establish a baseline, then train non-linear projectors on the graph classification dataset. Averaging F1 scores across two LLMs, results indicate that Vector-ICL enables LLMs to outperform the finetuned baseline model. This is a noteworthy achievement, as graph data is structurally very different from the text data that LLMs are trained on. The success here suggests that Vector-ICL can effectively translate graph structures into continuous representations that preserve relational information in a way that's interpretable to LLMs.

Brain fMRI Decoding and Classification

We pretrain projectors on the training set using next-token generation loss, then apply them to recover original text from brain fMRI signals in the test set. Results show LLMs can surpass random baselines by leveraging projected 200-dimensional fMRI PCA factors, with performance improving as context increases. This application to neuroscience data is particularly exciting, demonstrating Vector-ICL's potential in bridging neural activity and language understanding.

2.6 Analysis

2.6.1 The Impact of Encoder Quality on Vector-ICL Performance

We investigate the relationship between the intrinsic capabilities of encoders and their effectiveness when used with Vector-ICL for downstream tasks. To quantify the encoders' intrinsic abilities, we evaluate their performance on a text reconstruction task, which serves as a proxy for the amount of information preserved in the embeddings.

Our analysis focuses on text classification as the downstream task. We examine the correlation between encoder rankings on the reconstruction task and their corresponding rankings on the classification task. This analysis is performed across 5 datasets and 3 LLMs, resulting in

15 configurations.

The results of this analysis, presented in Fig. 2.4a, demonstrate a consistent positive correlation between an encoder's text reconstruction performance and its effectiveness in downstream classification tasks when used with Vector-ICL. Notably, in 4 of the 15 configurations, we observe a particularly strong correlation, with values approaching 1.

Our findings suggest that an encoder's performance on the text reconstruction task can serve as a reliable predictor of its potential effectiveness in downstream tasks when integrated with Vector-ICL. This insight could prove valuable for practitioners in selecting encoders for Vector-ICL.

2.6.2 Case study: what has been learned in the projections?

Fig. 2.4b provides a visualization of the normalized Euclidean distances between projected embeddings. Several key patterns emerge from this that offer insights into what the projector has learned.

Analysis of the numerical embedding distance matrix reveals key properties of our projection method. Embeddings for similar numbers cluster along the diagonal, indicated by lighter colors, demonstrating the preservation of local structure. Conversely, increasing distances from the diagonal, shown by darker colors, indicate effective separation of numerically distant values in the embedding space.

Another notable feature of the distance matrix is the block structure that emerges. The block structure reflects how numbers share similar digit patterns across the decimal places, from local to global blocks. It likely helps LLMs process numerical relationships more effectively, as it preserves the hierarchical nature of place-value notation.

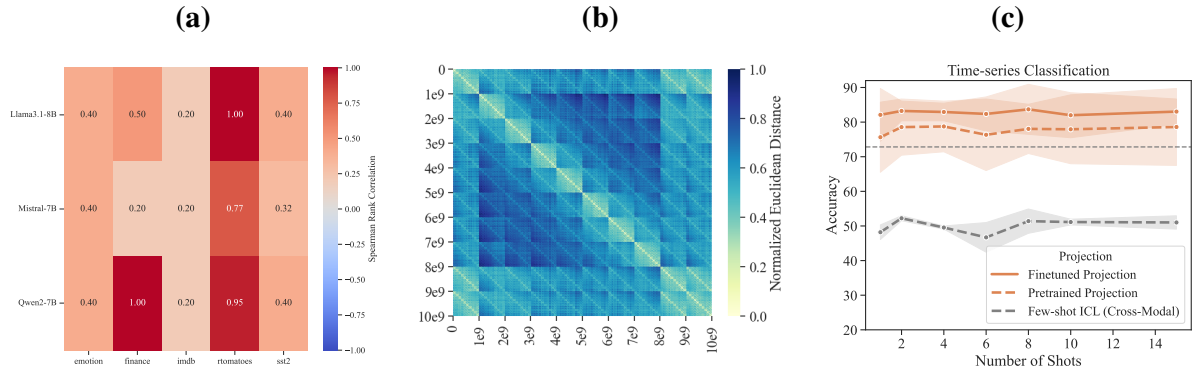


Figure 2.4. Key Insights from Encoders, Projections, and Synthetic Data Curation. (a) Correlation between encoders' text reconstruction performance and their downstream task effectiveness with Vector-ICL, suggesting information preservation ability predicts Vector-ICL performance. (b) Euclidean distance matrix of 1024 projected number embeddings (0 to 1e10) shows structured block-diagonal patterns, indicating meaningful numerical relationships are preserved. (c) Results from pretraining on our synthetic time-series QA dataset, which captures statistical properties through trend analysis, anomaly detection, and stability assessment. The curated QA pairs enable effective cross-modal pertaining.

2.6.3 Case study: can synthetic dataset be useful in cross-modal pre-training?

The cross-modal pretraining requires data-text dual pairs, which is not available in many cases. We explore how to create a synthetic dataset for such pretraining using time-series as an example. We designed a data curation pipeline that extracts meaningful statistical properties from time-series and converts them into natural language descriptions.

Our pipeline analyzes multiple statistical aspects of the time-series data. We perform trend analysis through linear regression to capture overall directional movements, anomaly detection using z-score analysis to identify unusual patterns, and temporal stability assessment via variability thresholds to characterize consistency.

For each identified property, we generate both descriptive statements and binary questions. For instance, given a time-series segment, we generate descriptive text like "The time-series shows an upward trend" or binary questions such as "Does this time-series contain any anomalies?"

Our experiments show that projectors pretrained on this synthetic corpus can effectively

generalize to new tasks, demonstrating the feasibility of creating synthetic data for cross-modal pretraining when natural data pairs are unavailable.

2.6.4 Detailed Experiment Setup

Text Reconstruction

We use two datasets for the text reconstruction task, Parallel Sentence Talks's English subset [Tiedemann, 2012] and Quora [Thakur et al., 2021].

Parallel Sentence Talks consist of sentences used in movie conversations, and Quora is built on a wide range of online questions. They represent short pieces of text that convey clear information. We aim to explore whether LLMs can decode the original message from the projected text embeddings.

We use the following prompt template:

Translate the text in brackets: ($\square$), translation: [Original Text]

We train the projectors on the training set of these two datasets and evaluate their performance on the test tests. We measure the reconstruction performance with the BLEU score [Papineni et al., 2002, Post, 2018].

Arithmetic and Function Regression

We created synthetic datasets of numerical data to pretrain our linear number projectors, experimenting with two configurations: one using 3-digit numbers and the other using 10-digit numbers. In Llama3.1-8B's tokenizer, 3-digit numbers are represented as single tokens, while in Mistral-7B, Qwen2-7B, and Yi-1.5-9B, numbers larger than 10 are split into multiple tokens. Consequently, 10-digit numbers consistently span multiple tokens across all models, which increases the complexity of performing arithmetic operations.

To represent the numbers, we use a concatenated-and-flattened one-hot vector encoding for each digit. For instance, a 3-digit number is represented as a 3×10 matrix (one hot per digit place), which is then flattened into a 30-dimensional vector. Similarly, a 10-digit number is represented as a 10×10 matrix, flattened into a 100-dimensional vector.

The pretraining involves two tasks. The first task is number reconstruction, we use the following prompt template, given the number is 128:

$$x = \square_x, \quad x \text{ equals to (digits): } \underline{128}$$

The second task is basic addition, we use the following prompt template, given the numbers are $x = 128$ $y = 256$, $a = 1$, $b = 1$:

$$x = \square_x,\ y = \square_y, \quad a * x + b * y \text{ equals to (digits): } \underline{384}$$

Here, a and b are randomly sampled from $[0, 1]$ with up to two decimal places, and the solution is the integer part of the sum.

For evaluation, we use a function regression task with a non-linear function: $f(x,y) = \sqrt{x} \cdot \sqrt{y}$. The LLM is given inputs x and y, along with the integer part of the output $f(x,y)$. The model is then tasked with predicting the output for new pairs of x and y. The prompt for in-context learning is structured as follows:

$$x = \square_x,\ y = \square_y, \quad \text{function}(x,y) \text{ equals to (digits): } \underline{\text{[Solution]}}$$

For few-shot ICL, the box tokens will be replaced with the actual numbers. We measure the function regression performance with the mean relative error, where the relative error is computed as the ℓ_1 difference divided by the ground truth value.

Text Classification

We use five datasets for the text classification task. For binary classification, we include IMDB [Maas et al., 2011], Rotten Tomatoes [Pang and Lee, 2005], and the Stanford Sentiment Treebank (SST2) [Socher et al., 2013]. For multi-class classification, we use the Emotion dataset [Saravia et al., 2018] and the Financial Phrasebank [Malo et al., 2014]. The binary classification datasets (IMDB, Rotten Tomatoes, and SST2) involve classifying movie reviews as positive or negative. The Emotion dataset classifies Twitter tweets into six categories: anger, fear, joy, love, sadness, and surprise. The Financial Phrasebank categorizes financial news into positive, negative, or neutral sentiments.

We use the following prompt:

(□)'s sentiment is [Input Class]

For few-shot ICL, the box tokens will be replaced with the actual text. For soft prompt tuning, we use 20 virtual tokens and train for one epoch over the entire training set. We measure the classification performance with accuracy on the test set.

Text Summarization

We use two datasets for the summarization task, XSum [Narayan et al., 2018] and the English subset of XLSum [Hasan et al., 2021]. They contain newspaper articles and their summaries.

We use the following prompt in ICL:

(□)'s summarization is: [Summary of the Input]

For few-shot ICL, the box token will be replaced by the article. We measure the performance using the RougeL score with the ground truth summary on the test sets.

Molecule Captioning

We use the Language + Molecules-24 (LPM24) dataset for the molecule captioning task, it was created for the task of molecule-language translation, and contains 161K pairs of molecule strings and their captions in the combined training and test set.

A sample molecule-caption pair looks like the following:

Molecule: Cc1c(Cl)cccc1-n1ccn2c(SCC(=O)c3ccccc3C(F)(F)F)nnc2c1=O

Caption: The molecule is a pain treatment that impacts inflammatory disease treatment.

And we use the following prompt for ICL:

(□)'s molecule caption is: [Caption of the Input Molecule]

For few-shot ICL, we replace the box token with the actual molecule string. We measure the performance using the BLEU score between the generated caption and the ground truth caption.

Brain fMRI Decoding and Classification

We analyze data from LeBel et al. 2022 and Tang et al. 2023, which consists of fMRI responses for 3 human subjects as they listen to 20+ hours of narrative stories from podcasts. We preprocessed the data following Benara et al. 2024, by converting the fMRI responses into a 200-dimensional output using principal components analysis and labeling 10-grams of the story text at 2-second intervals using an ensemble of LLMs.

The data was separated into train set and test set by holding out the same three podcast stories from the three human subjects. We use the same pretraining methodology as text to pretrain on the brain fMRI data. As the data comes in as segments of text and the recorded fMRI, we randomly sample 20% of the segments to be in fMRI form and projected into box tokens, and we impose next token generation loss on the rest 80%.

We evaluate the projectors by giving them randomly sampled context from the train set, that could come from different human subjects or from a different story, and ask them to decode segments from the test set. We use the following prompt in ICL in our decoding experiments:

> What is the English translation of the input?
>
> Input: □, Response: the input in English is
>
> [Text Corresponding to fMRI]

The random baseline is constructed by giving LLM the randomly sampled, shuffled text from the training set, and generating text according to it. We measure the performance using the RougeL score between the generated text and the ground truth.

We construct the classification questions around the properties of the underlying text, for example, “Does the sentence contain a proper noun?”, “”Does the input mention anything related to arguing?”. The ground truth is obtained via GPT4o [OpenAI, 2023] as binary labels. We use the following prompt in ICL in our classification experiments, using one of the example questions:

Does the input mention anything related to arguing?

Input: □, Response (Yes or No): according to the English text of the input, the answer is [Yes/No]

The random baseline is 50%, as we have downsampled and balanced the data. And we use accuracy as the performance metric.

Time-series

We use the Chronos [Ansari et al., 2024] time-series Transformers as the encoder. Chronos was pretrained on large scale time-series and is designed to generate the next segments of the time-series. It has proven effective on a wide range of time-series forecasting benchmarks. We take the output of the last time step from Chronos-base as the time-series representation.

We use two datasets for the time-series classification task, FordA, and FordB, they are also part of the UCR Time Series Classification Archive [Dau et al., 2019] ranging from 4000 to 5000 time-series for each dataset. We use the following prompt in ICL:

(□)'s class (positive, negative) is: [Input Class]

We construct a synthetic pretraining corpus through comprehensive analysis of additional time-series datasets from the UCR Time Series Classification Archive [Dau et al., 2019], specifically MoteStrain, TwoLeadECG, Wafer, PhalangesOutlinesCorrect, and Yoga. Our data curation framework includes multiple statistical approaches: trend identification via linear regression, anomaly detection using z-score analysis, and assessment of temporal stability through variability thresholds. To enhance corpus diversity, we incorporate binary (yes/no) questions targeting specific time-series characteristics. The resulting QA pairs create a rich textual representation that captures the underlying temporal and statistical properties of the data.

We use the base encoder with trained classification head as the baseline and we measure the prediction performance with accuracy.

Graphs

We use Graphormer [Ying et al., 2021] as the encoder model, specifically the one that was pretrained on large scale quantum chemistry graph datasets [Hu et al., 2021]. Since the down-stream task (ogbg-molhiv [Hu et al., 2020]) is a molecule property prediction dataset, and with strong class imbalance (3% positive classes), we finetune the encoder on the training set to provide meaningful baselines and embeddings. We take the output prior to the classification layer of the Graphormer as the graph embedding.

We use the ogbg-molhiv [Hu et al., 2020] dataset for the graph classification task. ogbg-molhiv is a molecule property prediction dataset consisting of a total 41.12K graphs with node features and edge attributes. It has a strong class imbalance of having around 3% positive class and 97% negative class.

Hence weighted sampling is adopted in the finetuning of both the baseline Graphormer and the embedding projector to yield meaningful predictions. We use the following prompt:

(□)'s class (positive, negative) is: [Input Class]

We use the finetuned Graphormer as our baseline and use the F1 score as the performance metric due to the significant class imbalance.

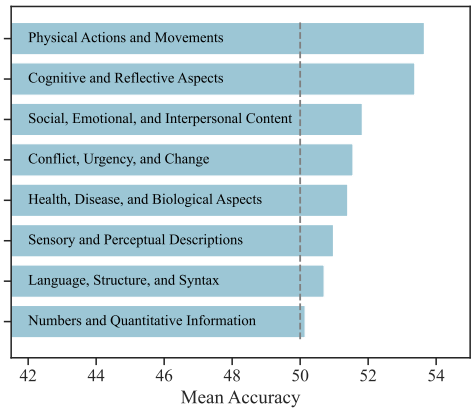


Figure 2.5. Analyzing LLM's understanding of projected brain fMRI embeddings after only unsupervised pretraining. We categorize the underlying questions related to the text and measure the mean accuracy for each category, highlighting the LLM's ability to interpret the embeddings, with only the next token prediction pretraining.

2.6.5 Case study: what information was perceived from projected brain fmri?

Fig. 2.5 illustrates the mean accuracy achieved in decoding different categories of brain activity based on fMRI data. The categories, ranging from ”Physical Actions and Movements” to ”Conflict, Urgency, and Change,” represent diverse cognitive and perceptual domains. The grouping and corresponding questions are listed in Table B.2.

The input data is noisy, and the projector is only trained with pretraining objectives, i.e., to predict the next piece of text given the current fMRI signal. We are surprised that with this pure unsupervised training, the LLM can still pick up meaningful signals from the projected embeddings. Notably, decoding tasks associated with “Physical Actions and Movements” and “Cognitive and Reflective Aspects” demonstrate higher mean accuracy, suggesting that these categories are more distinguishable based on the fMRI embeddings.

2.7 Discussion

Limitations and Future Directions

In this study, we explored a variety of settings: utilizing different encoders, LLM architectures, modalities, and datasets. Our results demonstrate that LLMs are capable of performing Vector-ICL on both language and non-language inputs. However, our experiments did not cover all possible combinations of these variables. There are still many unexplored areas, such as additional modalities, tasks, and encoder-decoder configurations, that could further benefit from Vector-ICL. Also, we only experimented with single-token encoders, while there exist encoders that produce variable-sized embeddings, that can potentially be more powerful and flexible. Analyzing how instruction tuning might affect the model’s ability to understand vector context would be beneficial as well. We leave this extensive exploration for future research to fully understand the broader applicability and limitations of our approach across diverse domains.

Conclusion

In this work, we explore whether large language models trained only on text can perform in-context learning on continuous vectors from different domains. Our findings suggest that LLMs can indeed understand and process continuous context via embedding projection. Simple linear projections are often sufficient, though for cross-modal tasks—such as those involving non-textual data like time-series or graphs—non-linear transformations may be required. In our experiments across various tasks and modalities, including text reconstruction, numerical function regression, text classification, summarization, molecule captioning, time-series classification, graph classification, and fMRI decoding, Vector-ICL often surpasses both few-shot ICL and task-specific model or tuning. We further conduct analyses and case studies, indicating the potential of LLMs to process vector representations beyond traditional token-based paradigms.

Acknowledgment

This work is sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

Chapter 2, in full, is a reprint of the paper: Zhuang, Yufan, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. "Vector-ICL: In-context Learning with Continuous Vector Representations." In The Thirteenth International Conference on Learning Representations (ICLR 2025). The dissertation author was the primary investigator and lead author of this paper.

Chapter 3

Self-Taught Agentic Long-Context Understanding

Answering complex, long-context questions remains a major challenge for large language models (LLMs) as it requires effective question clarifications and context retrieval. We propose Agentic Long-Context Understanding (AgenticLU), a framework designed to enhance an LLM's understanding of such queries by integrating targeted self-clarification with contextual grounding within an agentic workflow. At the core of AgenticLU is Chain-of-Clarifications (CoC), where models refine their understanding through self-generated clarification questions and corresponding contextual groundings. By scaling inference as a tree search where each node represents a CoC step, we achieve 97.8% answer recall on NarrativeQA with a search depth of up to three and a branching factor of eight. To amortize the high cost of this search process to training, we leverage the preference pairs for each step obtained by the CoC workflow and perform two-stage model finetuning: (1) supervised finetuning to learn effective decomposition strategies, and (2) direct preference optimization to enhance reasoning quality. This enables AgenticLU models to generate clarifications and retrieve relevant context effectively and efficiently in a single inference pass. Extensive experiments across seven long-context tasks demonstrate that AgenticLU significantly outperforms state-of-the-art prompting methods and specialized long-context LLMs, achieving robust multi-hop reasoning while sustaining consistent performance as context length grows.[1]

[1]Code and data is available at: https://github.com/EvanZhuang/AgenticLU.

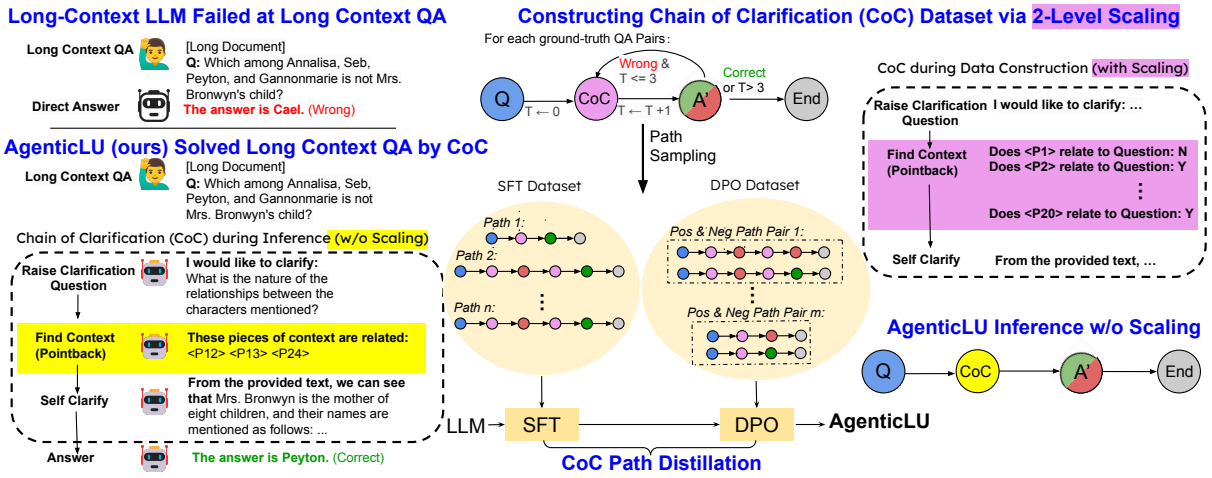


Figure 3.1. Overview of the AgenticLU pipeline: The model iteratively refines its understanding of long-context inputs through an agentic workflow. At each step, it raises self-clarifications, retrieves relevant context via the pointback mechanism, and updates its reasoning trace. The framework integrates CoC Path Construction to generate diverse reasoning paths, followed by two-stage fine-tuning (SFT and DPO) to enhance long-context understanding.

3.1 Introduction

Large language models have achieved notable milestones in natural language processing, demonstrating exceptional performance in tasks such as mathematical reasoning, code generation, and conversational understanding [OpenAI, 2023, DeepSeek-AI, 2025]. However, effectively comprehending and utilizing long-context inputs remains a major challenge. Complex queries often require models to retrieve multiple relevant pieces of information from extensive contexts and synthesize them coherently. While recent advancements have extended context windows to 128K and even 2M tokens [Dubey et al., 2024, Touvron et al., 2023b, Reid et al., 2024], these models still struggle to fully integrate and reason over large-scale contextual information. Recent studies [Liu et al., 2024, Gao et al., 2024] highlight a fundamental challenge in long-context understanding: the disparity between a model's nominal context size—the theoretical maximum input length—and its effective context window, the portion of the input the model actively utilizes for reasoning. This gap significantly impacts the understanding performance, limiting the model's ability to fully comprehend and integrate long-context information.

We introduce a novel framework AgenticLU to enhance long-context comprehension in LLMs. As illustrated in Fig. 3.1, the core of AgenticLU is **Chain-of-Clarifications** (CoC), a process where models enhance their understanding by generating clarification questions, retrieving relevant information from the long context and answering their own clarification questions based on the gathered evidence. Rather than relying on a direct response, CoC helps models refine their reasoning iteratively, resolving uncertainties along the way. We structure the framework into the following two stages.

CoC Path Construction.

To collect reliable CoC understanding path, we structure data collection as a tree search, where each CoC step represents a node. We leverage extended inference time to determine the effective clarification questions to ask and the relevant evidence to retrieve. With a search depth of three and a branching factor of eight, AgenticLU successfully retrieves 97.8% of the correct answers in NarrativeQA [Kočiský et al., 2018], demonstrating its capability to tackle complex questions that require multi-step reasoning over long-context inputs.

CoC Path Distillation.

Once the dataset is collected from the tree-search process, we train the model to generate effective clarifications and contextual groundings in a single pass, eliminating the need for scaling at inference time. This is achieved by distilling these collected paths into LLMs through supervised finetuning (SFT) and direct preference optimization (DPO) [Rafailov et al., 2024], effectively amortizing the computational cost from inference to training.

Our method AgenticLU significantly improves model's long-context understanding capabilities without relying on laborious human annotations or stronger teacher models for data generation. Instead, the base model's self-generated CoC paths enables it to teach itself to process long-context inputs more effectively. This approach harnesses the model's inherent long-context capabilities—previously only accessible through an additional LLM agent—allowing it to independently refine its reasoning and retrieval processes. Empirically, we demonstrate

that AgenticLU consistently boosts performance across a set of question-answering tasks up to 128K tokens, outperforming both prompting-based approaches and other long-context-finetuned LLMs. By integrating self-clarification and context grounding in an agentic manner, we take a step further toward enabling LLMs to comprehend long contexts.

3.2 Related Work

Challenges in Long Context Understanding

LLMs struggle with long contexts despite supporting up to 2M tokens [Dubey et al., 2024, Reid et al., 2024]. The "lost-in-the-middle" effect [Liu et al., 2024] and degraded performance on long-range tasks [Li et al., 2023a] highlight these issues. To address this, ProLong [Gao et al., 2024] finetunes base models on a large, carefully curated long-context corpus. While this approach improves performance on long-range tasks, it comes at a significant cost, requiring training with an additional 40B tokens and long-input sequences.

Inference-time Scaling for Long-Context

The Self-Taught Reasoner (STaR) framework [Zelikman et al., 2022] iteratively generates rationales to refine reasoning, with models evaluating answers and finetuning on correct reasoning paths. Wang et al. [2024b] introduced Model-induced Process Supervision (MiPS), automating verifier training by generating multiple completions and assessing accuracy, boosting PaLM 2's performance on math and coding tasks. Li et al. [2024] proposed an inference scaling pipeline for long-context tasks using Bayes Risk-based sampling and fine-tuning, though their evaluation is limited to shorter contexts (10K tokens) compared to ours (128K tokens).

Agentic Workflow for Long-Context

Agentic workflows [Yao et al., 2022] enable LLMs to autonomously manage tasks by generating internal plans and refining outputs iteratively. The LongRAG framework [Zhao et al., 2024c] enables an LLM and an RAG module to collaborate on long-context tasks by breaking down the input into smaller segments, processing them individually, and integrating the results

to form a coherent output. Chain-of-Agents (CoA) [Zhang et al., 2024b] tackles long-context tasks through decomposition and multi-agent collaboration. In CoA, the input text is divided into segments, each handled by a worker agent that processes its assigned portion and communicates its findings to the next agent in the sequence. Unlike these, our approach employs a single LLM that orchestrates its own reasoning and retrieval without relying on multiple components. By dynamically structuring its process and iteratively refining long-context information, our model reduces complexity while maintaining efficiency.

3.3 The Context Size Gap

State-of-the-art LLMs have made strong claims about their context lengths, supporting hundreds of thousands of input tokens. However, recent studies [Gao et al., 2024, Yen et al., 2024, Shang et al., 2024] have shown that the *effective* context size of an LLM (the length over which it can reliably perform tasks such as information retrieval and complex reasoning) often diverges from its claimed, or *nominal*, context length.

To illustrate this gap, we evaluate Llama3.1-8B-Instruct, which supports a 128K-token context, on the HotPotQA dataset to test multi-hop QA performance at various input lengths (8K, 16K, 32K, 64K, and 128K). We artificially expand the input by adding irrelevant context and measure the accuracy of its answers using GPT-4o as a judge. As shown in Fig. 3.2, The model's performance degrades substantially as increasing context length, demonstrating the discrepancy between nominal and effective context sizes.

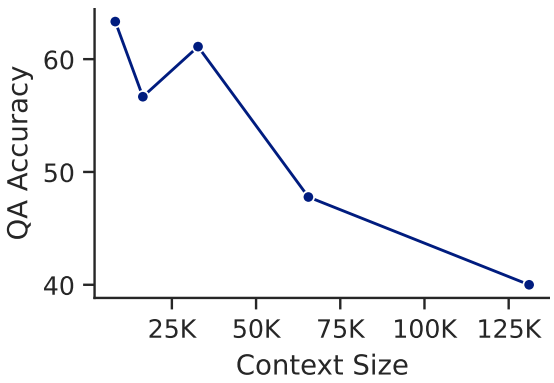


Figure 3.2. Effective context size is smaller than nominal context size. Performance of Llama3.1-8B-Instruct (advertised 128K-token context) on the HotPotQA dataset drops sharply as input length increases (8K, 16K, 32K, 64K, 128K), illustrating the gap between nominal and effective context capacities.

While expanding nominal context capacity

is undoubtedly important, we argue that it is not sufficient for solving all long-context problems. By analogy with computer memory, simply having more capacity does not guarantee efficient or accurate computation; one must also manage the "loading" of relevant information in and out of this memory. Therefore, we propose an agentic workflow aimed at helping LLMs process and interpret extended contexts more intelligently.

3.4 Chain-of-Clarifications Workflow

Our approach centers on enhancing long-context comprehension through an iterative, self-refining process that blends inference-time scaling with agentic reasoning. We coin this agentic workflow Chain-of-Clarifications (CoC). In this section, we detail its key components, including the self-clarification process and the pointback mechanism, as illustrated in Fig. 3.1.

Our proposed CoC framework is designed to mitigate the gap between nominal and effective context sizes in large language models. Rather than processing the entire long context and potentially multi-hop questions in a single pass, our methodology decomposes the task into a sequence of targeted sub-tasks. At each CoC step, the model autonomously:

- **Generates clarifying questions** by identifying areas of the long input that require further elaboration or are prone to misinterpretation.
- **Pointbacks to relevant context** by using a pointback mechanism that highlights critical segments of the context by naming the index of relevant paragraphs. In the data collection phase, this is done by iteratively querying the LLM about the relevance of each paragraph with respect to the question. After training, the model is finetuned to generate the related paragraph indexes directly in a single pass.
- **Answers clarifying questions** by integrating highlighted context into consideration to build a more accurate and contextually grounded understanding of the long document.

- **Answers the original question** by combining all newly gathered clarifications, the model attempts to generate a valid answer to the original question.

It is important to note a key distinction between CoC path generation during data collection and the actual task deployment of the agentic workflow. In the data generation phase, we prompt the LLM to iteratively process each chunk of input text along with its self-generated clarifying questions, ensuring accurate retrieval of relevant context. During training, rather than relying on repeated inference calls, we finetune the model to directly generate the indexes of relevant paragraphs using pointback examples, effectively amortizing the computational cost into training. This enables the model to internalize the retrieval process, allowing it to dynamically synthesize relevant clarifications and contextual references at inference time without requiring extensive additional prompting.

3.5 Data Generation & Model Training

Dataset

We use the NarrativeQA [Kočiský et al., 2018] dataset to facilitate long-context QA and generate agentic workflow traces with 14.7K QA pairs in the training set. NarrativeQA is designed for reading comprehension over narrative texts, such as books and movie scripts, where each example includes a full story and a set of corresponding QA pairs. This dataset emphasizes deeper reasoning and long-context understanding, as many questions require synthesizing information from multiple parts of the narrative rather than focusing solely on particular local context. Its relatively long passages make NarrativeQA particularly suitable for testing and refining agentic reasoning in large language models, as the answers often depend on weaving together details spanning the entire text.

Base Model

Our base model is *Llama3.1-8B-Instruct* [Dubey et al., 2024], an 8-billion-parameter instruction-tuned Llama model. This model is built on the same transformer architecture as

Llama3, but with additional fine-tuning data to improve its performance on multi-turn dialogue and instruction-following tasks.

3.5.1 CoC Path Construction

We employ a test-time scaling approach to generate CoC paths. For each question, we construct a tree of search paths where each node represents a distinct clarification question posed by the LLM.

In our experiments, we use a branching factor of 8 at each depth and select the most promising trace based on an evaluation score that combines:

- **Semantic similarity**, measured by the RougeL [Lin, 2004] score relative to the ground truth.
- **Discrete correctness**, evaluated by a binary verification using GPT4o-mini.

In the data construction process, the relevant context is found by iteratively querying the LLM about the relevance of all chunked passages. Here we use 512 as the chunk size. This process is compute-intensive but only happens in data collection. After the training, the LLM will directly generate the paragraph numbers of the relevant context as shown in the lower right of Fig. 3.1.

For most long-context tasks, a single clarification question suffices because the required reasoning is not highly complex. 92% of the questions in our experiments are resolved correctly with just one round of clarification. More challenging tasks may require multiple rounds of clarification: two rounds resolve 53% of the remaining 8%, and three rounds resolve 35% of the remaining 4%. Because of the exponentially increasing cost—and given that 97.4% of the training questions are already solved—we limit the maximum depth of our inference scaling to 3.

The statistics of the collected dataset are shown in Table 3.1. The total number of conditional generation tokens that the LLM trained on is 17M tokens, with input that has an

Table 3.1. Statistics of the generated traces dataset used in finetuning derived from NarrativeQA. We left out 11.9K traces for validation.

Data	#
Num of Traces	107,550
Avg Context Length	67,812
Avg Chosen Response Length	165
Avg Rejected Response Length	164
Total Generation Tokens	17M

average length of 67K and a max length of 128K tokens.

3.5.2 CoC Path Distillation

We employ a two-stage finetuning recipe: Supervised Fine-Tuning (SFT) followed by Direct Preference Optimization (DPO) [Rafailov et al., 2024], to convert our base model into a long-context understanding agent. The dataset statistics is described in Table 3.1, with input length up to 128K tokens.

Supervised Fine-Tuning

In the first phase, we finetune *Llama3.1-8B-Instruct* using the generated CoC paths. Each training example includes (1) the full context from NarrativeQA, (2) the question, and (3) the step-by-step reasoning trace leading to the final answer. By exposing the model to these traces, we encourage it to internalize multi-step reasoning strategies and context grounding for the long-context inputs. The SFT stage uses a standard cross-entropy loss on the next-token prediction task, ensuring the model learns how to produce consistent and complete reasoning sequences.

Direct Preference Optimization

In the second phase, we apply Direct Preference Optimization to further refine the model's output quality. To create preference pairs, we sample incorrect workflow traces as negative examples with using GPT4o-mini as the judge for answer correctness from the test-time scaling. DPO explicitly optimizes the model to generate higher-ranked responses more frequently, thus

Table 3.2. Performance difference of AgenticLU and its base, Llama3.1-8B-Instruct (δ =AgenticLU-8B minus Llama3.1-8B), on long context (the 128K tasks) and short-context benchmarks (6 regular tasks including ARC, GSM8K, and MMLU), the details of the short-context performance can be found in Sec. C.2. Scores represent accuracy, with AgenticLU demonstrating significantly improved performance across long-context tasks with minimal effect on regular task performance.

Model	**Short Avg**	HotpotQA	Natural Questions	TriviaQA	PopQA	NarrativeQA	InfiniQA	InfiniChoice	**Long Avg**
Llama3.1-8B	**62.3**	40.0	56.1	80.6	56.1	38.0	48.0	55.0	**53.4**
AgenticLU (δ)	**-0.6**	+31.1	+21.7	+7.7	+9.4	+18.0	+2.0	+13.0	**+14.7**

aligning the agent's outputs with desirable characteristics, such as clarity, correctness, and coherence. This stage ensures that even among valid reasoning paths, the model learns to prioritize the most instructive reasoning.

The details for the two-phase training are listed in Sec. C.1.

3.6 Evaluation

In this section, we assess our method AgenticLU using a suite of evaluation tasks drawn from the HELMET long-context benchmark [Yen et al., 2024]. Our experiments focus on testing models' ability to retain, process, and reason over extended contexts ranging from 8K to 128K tokens.

3.6.1 Tasks and Metrics

We evaluate our models and baselines on the Helmet [Yen et al., 2024] long-context evaluation benchmark's retrieval-augmented generation (RAG) and long-range QA (LongQA) tasks ranging from 8K, 16K, 32K, 64K, to 128K.

We use GPT-4o as the judge for answer correctness, with the prompt template shown in Sec. C.5. We report accuracies for all datasets.

The RAG test suite includes: (1) **HotpotQA** [Yang et al., 2018], a multi-hop reasoning dataset over Wikipedia; (2) **Natural Questions** [Kwiatkowski et al., 2019], real user queries with Wikipedia-based short and long answers; (3) **TriviaQA** [Joshi et al., 2017], a large-scale trivia

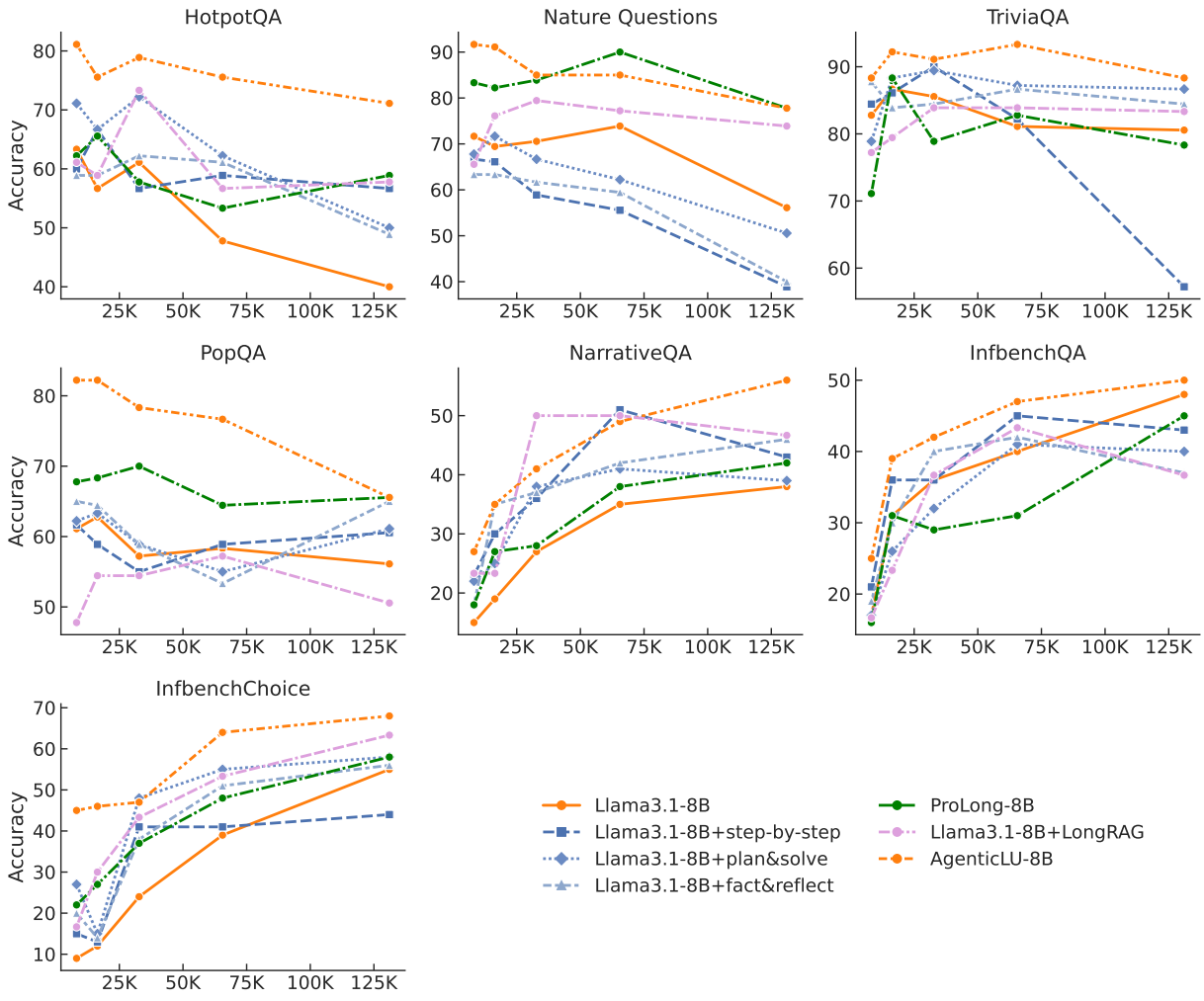


Figure 3.3. Main results on 7 long-context tasks across context lengths from 8K to 128K. Our AgenticLU-8B (dotted orange) achieves significant improvements on *all* tasks over our base model Llama3.1-8B (solid orange). We also compare with the prompting methods (Step-by-Step, Plan-and-Solve, Fact-and-Reflect, LongRAG) and the state-of-the-art ProLong-8B model. AgenticLU-8B consistently maintains strong performance across most tasks and context lengths.

dataset with question-answer pairs linked to evidence documents; (4) **PopQA** [Mallen et al., 2022], a dataset testing model memorization with fact-based questions from popular culture.

The LongQA test suite includes: (1) **NarrativeQA** [Kočiský et al., 2018], a reading comprehension dataset with Wikipedia summaries and story-based Q&A; (2) **InfiniteBench QA** [Zhang et al., 2024a], a long-range QA benchmark requiring reasoning over extended contexts; (3) **InfiniteBench Multiple-Choice** [Zhang et al., 2024a], a multiple-choice variant of

the previous evaluating reading comprehension over long documents.

For the four RAG tasks, each question is put alongside a set of relevant contexts, and the overall input length is increased by appending irrelevant context. Consequently, these tasks become strictly more difficult as the context window expands. In contrast, for the three LongQA tasks, the relevant context may not appear in the truncated input (the first 8K, 16K, or 128K tokens). Hence, performance might improve at longer input lengths simply because the necessary information becomes available only after including more tokens.

3.6.2 Baselines

We compare AgenticLU against a diverse set of strong baselines representing different approaches for handling long-context tasks. Our comparisons include two main categories.

Under prompting methods we consider techniques that require no additional model training. In particular, we evaluate (a) the chain-of-thought approach [Kojima et al., 2022], which encourages models to decompose complex questions into intermediate reasoning steps; (b) fact-and-reflection prompting [Zhao et al., 2024d], which iteratively verifies and refines factual claims to enhance consistency; (c) plan-and-solve prompting [Wang et al., 2023], where the model first outlines a high-level plan before sequentially executing it to address structured reasoning tasks; and (d) LongRAG [Zhao et al., 2024a] where a hybrid RAG system is used to retrieve relevant context to generate global summaries and local details [2].

In the fine-tuning category, we focus on models that have been specifically adapted for extended context data. For a substantial comparison, we employ Prolong-8B-512K [Gao et al., 2024]—a model based on the Llama3 8B architecture that has been further trained on an additional 40B tokens of long-context data.

3.6.3 Main Results

The performance of AgenticLU and baseline models is shown in Fig. 3.3.

[2]Note that LongRAG provided finetuned models as well. But the SFT-ed Llama3-8B only supports 8K context length. Thus we did not include it in our comparison.

Self-clarification significantly improves multi-hop reasoning.

AgenticLU-8B consistently surpasses other methods in HotpotQA. By iteratively refining its understanding, resolving ambiguities, and verifying intermediate steps, the model achieves higher accuracy, particularly as context length increases.

Robust performance across diverse datasets.

Unlike baseline models, AgenticLU-8B maintains consistently strong performance across RAG and LongQA benchmarks, demonstrating its ability to adapt effectively to different long-context tasks.

Reduced performance degradation with longer contexts.

While most models experience significant accuracy drops as context length increases, AgenticLU-8B remains stable. Its self-clarification and pointback mechanisms effectively filter noise from irrelevant information, allowing the model to extract and prioritize essential evidence.

Fine-tuning vs. prompting trade-offs.

While structured prompting techniques like *plan-and-solve* improve short-context reasoning, they struggle with extreme context lengths (e.g., 128K tokens). In contrast, AgenticLU-8B, through targeted finetuning with self-clarification and pointback, maintains robust long-context reasoning without relying on complex prompting strategies. Although ProLong-8B, another finetuned model, achieves strong results, it comes with significantly higher training costs. AgenticLU-8B, by contrast, is more data-efficient and generalizes better to novel tasks, making it a more practical and effective solution for long-context reasoning.

Overall, these results underscore the effectiveness of AgenticLU-8B in tackling long-context understanding challenges. The integration of self-clarification plays a crucial role in improving grounding, reasoning, and comprehension in long-context settings.

Table 3.3. We evaluate the performance of adding additional self-clarification and contextual grounding rounds at inference time. The gain from self-clarification is close to optimal at the initial round.

Model	HotpotQA	NaturalQ	PopQA	TriviaQA	Avg
Llama-3.1-8B	40.0	56.1	56.1	80.6	58.2
AgenticLU-8B	71.1	77.8	65.5	88.3	75.7
(w/ 2 rounds)	71.1	76.7	67.2	91.7	76.7
(w/ 3 rounds)	75.5	78.8	68.3	91.1	78.4

3.6.4 Performance on Short-Context Tasks

To demonstrate that our fine-tuning process preserves the model's general capabilities while enhancing long-context understanding, we evaluated the finetuned model on a diverse set of standard benchmarks. These include elementary and advanced reasoning tasks ARC Easy and ARC Challenge [Clark et al., 2018], mathematical problem-solving GSM8K [Cobbe et al., 2021], MathQA [Amini et al., 2019], and broad knowledge assessment MMLU [Hendrycks et al., 2021b,a], MMLU-Pro [Wang et al., 2024a].

We report the average performance across short-context tasks in Table 3.2, and each individual task result can be found in Sec. C.2. We find that the short-context performance is well preserved, demonstrating that AgenticLU's core reasoning and problem-solving abilities remain strong and are not compromised by the significant improvements to its long-context understanding powers.

3.7 Analysis

In this section, we take a closer look at how each part of our approach affects long-context understanding and retrieval. Specifically, we study three main questions: (1) Can the finetuned system benefit from multi-round CoC? (2) Does adding clarifications and pointing back to the original document help the model understand and utilize the context more accurately? (3) How much additional compute overhead does AgenticLU add to the process?

Table 3.4. We test the agentic workflow with AgenticLU-8B when taking out the self-clarification steps and the contextual grounding (pointback) step. The tasks are with 128K context length.

Model	HotpotQA	NaturalQ	PopQA	TriviaQA	Avg
Llama-3.1-8B	40.0	56.1	56.1	80.6	58.2
AgenticLU-8B	71.1	77.8	65.5	88.3	75.7
(w/o Clarification)	57.8	56.7	55.5	78.3	62.1
(w/o Pointback)	53.3	59.4	52.7	83.3	62.2

3.7.1 How many rounds of CoC are needed?

Setup.

We add additional rounds of reasoning in the evaluation and see if the LLM can benefit from multi-rounds of reasoning at test-time.

Analysis.

The results, presented in Table 3.3, indicate that additional rounds of agentic reasoning do provide performance improvements.

This suggests that while significant benefits of self-clarification are achieved in the first round, additional rounds still contribute to further improvements. One possible explanation is the nature of our dataset: approximately 92% of the questions are resolved within a single round of clarification. However, for the remaining cases, extended reasoning allows the model to refine its understanding, leading to measurable gains in performance with more clarification and reasoning.

3.7.2 Do Self-Clarifications and Pointback Help in Long-Context Understanding?

Setup.

To evaluate the impact of each component in our agentic workflow, we compare the full AgenticLU-8B model against two variants: one without the self-clarification step and another without the contextual grounding (*pointback*) step. We use the four RAG datasets with 128K context length as the evaluation benchmark, and compare the performance alongside the original model.

Table 3.5. Performance Overhead Comparison between direct answering baseline and AgenticLU.

Metric	Baseline	AgenticLU
Runtime Overhead	100%	101.93%
Avg Tokens Generated in One Round	76.28	1205.38

Analysis.

Table 3.4 shows the results on four QA benchmarks with a 128K context length. Removing self-clarification leads to an absolute performance drop of at least 10 points across most tasks (e.g., from 71.1% to 57.8% on HotpotQA), confirming that the model benefits from clarifying its own uncertainties when the context is long. Meanwhile, omitting pointback yields degenerate results, indicating that pinpointing relevant information at each stage is crucial for long-context QA. Overall, these findings highlight the importance of both clarifications and context-grounding to maximize retrieval accuracy and robustness in lengthy documents.

3.7.3 How much additional compute cost does AgenticLU impose in generation?

Since additional generation steps are introduced in the QA process, we assess the overhead in inference time. Naïvely, long-context inference and multi-round conversations could significantly amplify compute costs. However, by leveraging prefix caching to store computed KV caches, the additional cost scales linearly with the number of newly generated tokens rather than exponentially.

To quantify this overhead, we conduct a runtime evaluation on 100 queries with a 128K context size. The results, summarized in Table 3.5, demonstrate that the additional computational overhead remains minimal when using prefix caching.

3.8 Discussion

In this work, we introduce Agentic Long-Context Understanding (AgenticLU), a framework designed to enhance large language models' ability to process and reason over long-context

inputs with self-generated data. By incorporating an agentic workflow (CoC) that dynamically refines model reasoning through self-clarifications and contextual grounding, AgenticLU significantly improves LLM's long context understanding capabilities.

Through a combination of trace data collection and two-stage post-training, our approach enables models to autonomously explore multiple reasoning paths, distill the most effective clarification strategies, and improve their understanding of lengthy documents. Extensive evaluations on long-context benchmarks demonstrate that AgenticLU outperforms existing prompting techniques and finetuned baselines, maintaining strong performance across context lengths up to 128K tokens. Additionally, ablation studies confirm that self-clarification and pointback mechanisms play a crucial role in improving retrieval and reasoning over long-contexts.

Limitations

Despite its effectiveness in long-context reasoning, AgenticLU has notable limitations. One key drawback is its inability to autonomously determine when to stop multi-round reasoning. While additional rounds of self-clarification can improve performance, the model follows a fixed number of reasoning steps rather than dynamically assessing when further refinement is necessary. This can lead to inefficiencies, where the model either stops too early, missing potential improvements, or continues reasoning unnecessarily, expending computational resources without significant gains.

Developing a fully agentic mechanism remains an open challenge. Ideally, the model should assess its confidence in an intermediate response and decide whether further clarification is needed. Future work should explore approaches that enable AgenticLU to regulate its reasoning depth dynamically, optimizing both efficiency and performance.

Acknowledgment

This work is sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, Sponsored Research Projects from companies like Cisco and eBay, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

Chapter 3, in full, is a reprint of the paper: Zhuang, Yufan, Xiaodong Yu, Jialian Wu, Ximeng Sun, Ze Wang, Jiang Liu, Yusheng Su, Jingbo Shang, Zicheng Liu, and Emad Barsoum. "Self-Taught Agentic Long Context Understanding." In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5525-5537. 2025. The dissertation author was the primary investigator and lead author of this paper.

Chapter 4

Summary and Future Work

The central contribution of this dissertation is a unified view of how language models learn to learn. Conceptually, this work challenges the view that learning to learn is tied to in-context learning, and is limited to discrete text. The three investigations show that it decomposes into at least three distinct but related capabilities internalizing algorithms, generalizing across representations, and constructing bootstrapped reasoning, each of which can be studied, extended, and improved independently. This decomposition offers a clearer understanding for what language models can do as meta-learners.

Practically, the results point toward a design philosophy for adaptive AI systems. Models can serve as algorithm designers rather than algorithm executors; their adaptation machinery can be reused across domains without domain-specific architectures; and self-generated chain-of-thoughts can substitute for human annotation in complex tasks while scaling with inference-time computation.

Looking forward, the most important open question this work raises is whether models can learn as we do: interact with the world, accumulate knowledge, and self-improve with its own thinking. Understanding the meta-learning behaviors that already exist in current language models, as this dissertation has attempted to do, is a necessary foundation for that longer-term goal.

Appendix A

Supplementary Material for Chapter 1

A.1 Model Hyperparameters

We list MetaTree's hyperparameters in Table A.1. We note that in the following ablation studies, the base and ablation models are trained with fewer steps (2M steps instead of 4M steps) due to the compute resource limit.

Table A.1. Hyperparameters for MetaTree training.

Hyperparameter	**Value**
Number of Hidden Layers	12
Number of Attention Heads	12
Hidden Size	768
Number of Parameters	149M
Learning Rate	5e-5
Learning Rate Schedule	Linear Decay
Optimizer	AdamW
β_1	0.9
β_2	0.999
Training dtype	bf16
Number of Features	10
Number of Classes	10
Block Size	256
Tree Depth	2
σ	5e-2
Number of Warmup Steps	1000
Number of Training Steps	4,000,000
Steps in Phase 1 (GOSDT)	1,000,000
Steps in Phase 2 (GOSDT+CART)	3,000,000
Batch Size	128

A.2 Datasets

Table A.2. List of 91 datasets used in MetaTree's evaluation.

Dataset	Entries	Dim.	Class
mfeat fourier	2000	76	10
mfeat zernike	2000	47	10
mfeat morphological	2000	6	10
mfeat karhunen	2000	64	10
page blocks	5473	10	5
optdigits	5620	64	10
pendigits	10992	16	10
waveform 5000	5000	40	3
Hyperplane 10 1E 3	1000000	10	2
Hyperplane 10 1E 4	1000000	10	2
pokerhand	829201	5	10
RandomRBF 0 0	1000000	10	5
RandomRBF 10 1E 3	1000000	10	5
RandomRBF 50 1E 3	1000000	10	5
RandomRBF 10 1E 4	1000000	10	5
RandomRBF 50 1E 4	1000000	10	5
SEA 50	1000000	3	2
SEA 50000	1000000	3	2
satimage	6430	36	6
BNG labor	1000000	8	2
BNG breast w	39366	9	2
BNG mfeat karhunen	1000000	64	10

Continued on next page

Table A.2. List of 91 datasets used in MetaTree's evaluation. (continued)

Dataset	Entries	Dim.	Class
BNG bridges version1	1000000	3	6
BNG mfeat zernike	1000000	47	10
BNG cmc	55296	2	3
BNG colic ORIG	1000000	7	2
BNG colic	1000000	7	2
BNG credit a	1000000	6	2
BNG page blocks	295245	10	5
BNG credit g	1000000	7	2
BNG pendigits	1000000	16	10
BNG cylinder bands	1000000	18	2
BNG dermatology	1000000	1	6
BNG sonar	1000000	60	2
BNG glass	137781	9	7
BNG heart c	1000000	6	5
BNG heart statlog	1000000	13	2
BNG vehicle	1000000	18	4
BNG hepatitis	1000000	6	2
BNG waveform 5000	1000000	40	3
BNG zoo	1000000	1	7
vehicle sensIT	98528	100	2
UNIX user data	9100	1	9
fri c3 1000 25	1000	25	2
rmftsa sleepdata	1024	2	4
JapaneseVowels	9961	14	9

Continued on next page

Table A.2. List of 91 datasets used in MetaTree's evaluation. (continued)

Dataset	Entries	Dim.	Class
fri c4 1000 100	1000	100	2
abalone	4177	7	2
fri c4 1000 25	1000	25	2
bank8FM	8192	8	2
analcatdata supreme	4052	7	2
ailerons	13750	40	2
cpu small	8192	12	2
space ga	3107	6	2
fri c1 1000 5	1000	5	2
puma32H	8192	32	2
fri c3 1000 10	1000	10	2
cpu act	8192	21	2
fri c4 1000 10	1000	10	2
quake	2178	3	2
fri c4 1000 50	1000	50	2
fri c0 1000 5	1000	5	2
delta ailerons	7129	5	2
fri c3 1000 50	1000	50	2
kin8nm	8192	8	2
fri c3 1000 5	1000	5	2
puma8NH	8192	8	2
delta elevators	9517	6	2
houses	20640	8	2
bank32nh	8192	32	2

Continued on next page

Table A.2. List of 91 datasets used in MetaTree's evaluation. (continued)

Dataset	Entries	Dim.	Class
fri c1 1000 50	1000	50	2
house 8L	22784	8	2
fri c0 1000 10	1000	10	2
elevators	16599	18	2
wind	6574	14	2
fri c0 1000 25	1000	25	2
fri c2 1000 50	1000	50	2
pollen	3848	5	2
mv	40768	7	2
fried	40768	10	2
fri c2 1000 25	1000	25	2
fri c0 1000 50	1000	50	2
fri c1 1000 10	1000	10	2
fri c2 1000 5	1000	5	2
fri c2 1000 10	1000	10	2
fri c1 1000 25	1000	25	2
visualizing soil	8641	3	2
socmob	1156	1	2
mozilla4	15545	5	2
pc3	1563	37	2
pc1	1109	21	2

Appendix B

Supplementary Material for Chapter 2

B.1 Text classification and text summarization config in tables

Llama3.1-8B is used as the common LLM and NV-Embed-v1 is used as the text encoder for Vector-ICL. The ICL and soft-prompt methods are supplied with up to 50 shots as context. The soft-prompt tokens are trained over the entire training set.

B.2 Additional Experimental Results

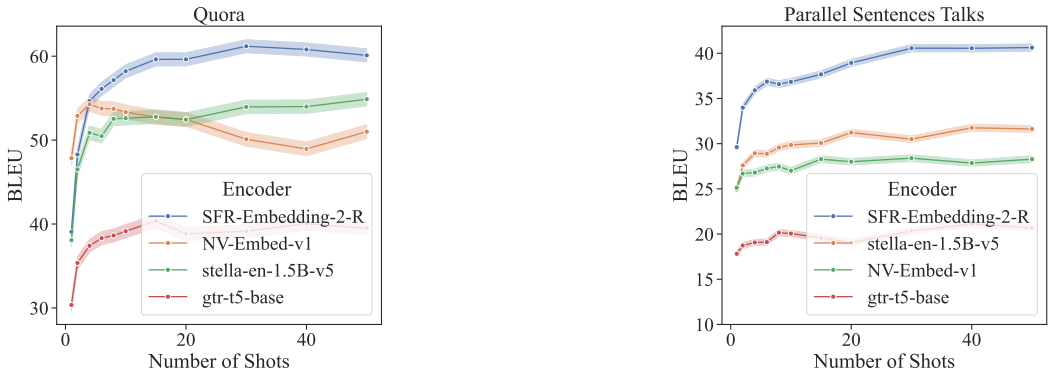


Figure B.1. Text Reconstruction

Table B.1. Hyperparameters for V-ICL training.

Hyperparameter	**Value**
Learning Rate	1e-3
Learning Rate Schedule	Cosine Annealing
Optimizer	AdamW
β_1	0.9
β_2	0.999
Training dtype	bf16
Batch Size	128
Generation Temperature	2e-1

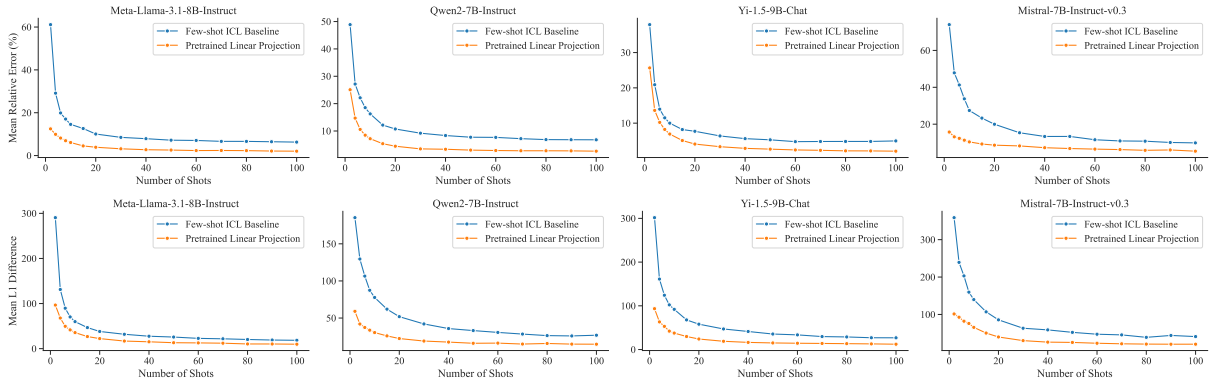


Figure B.2. Function Regression - 10 digits (upper) and 3 Digits (lower)

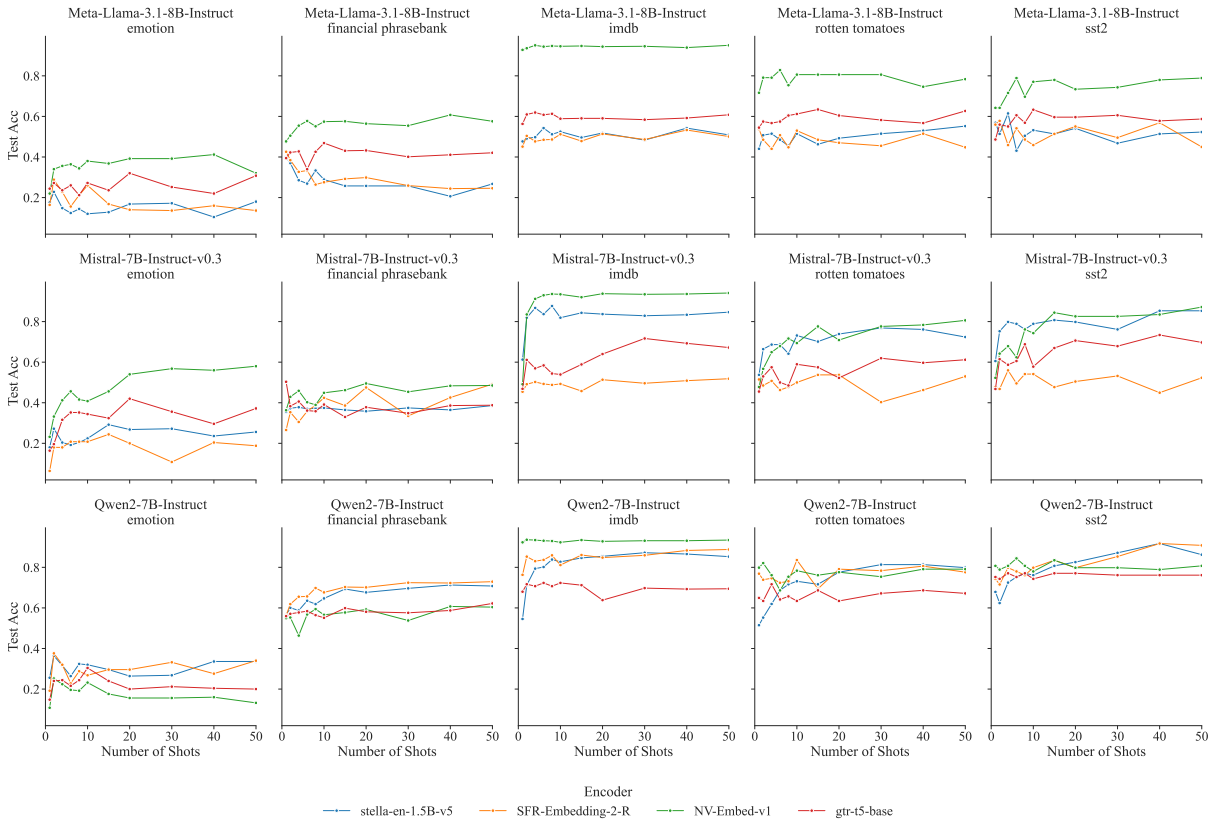


Figure B.3. Text Classification - Pretrained Projectors

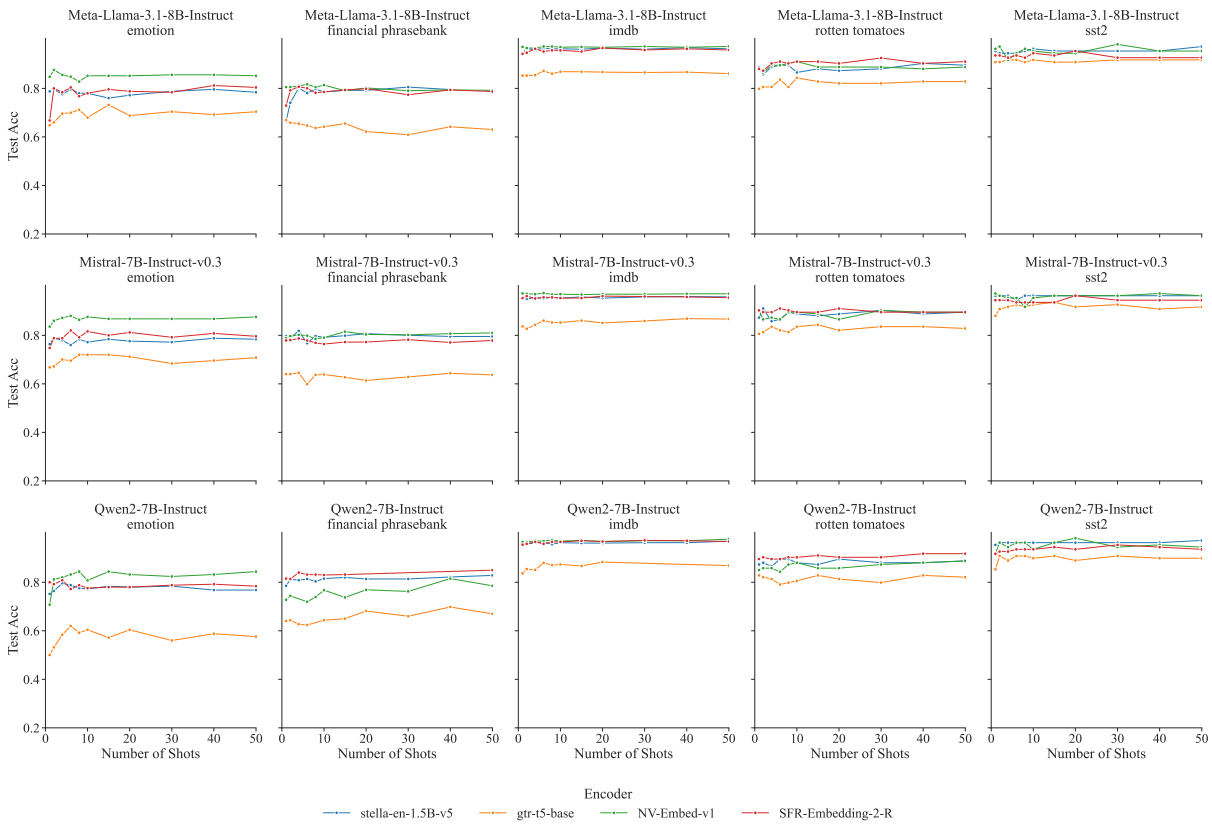


Figure B.4. Text Classification - Finetuned Projectors

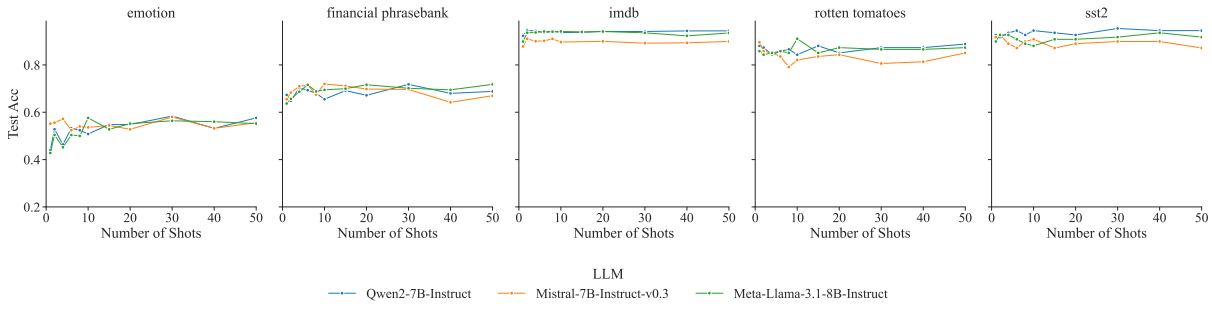


Figure B.5. Text Classification - Few-shot ICL

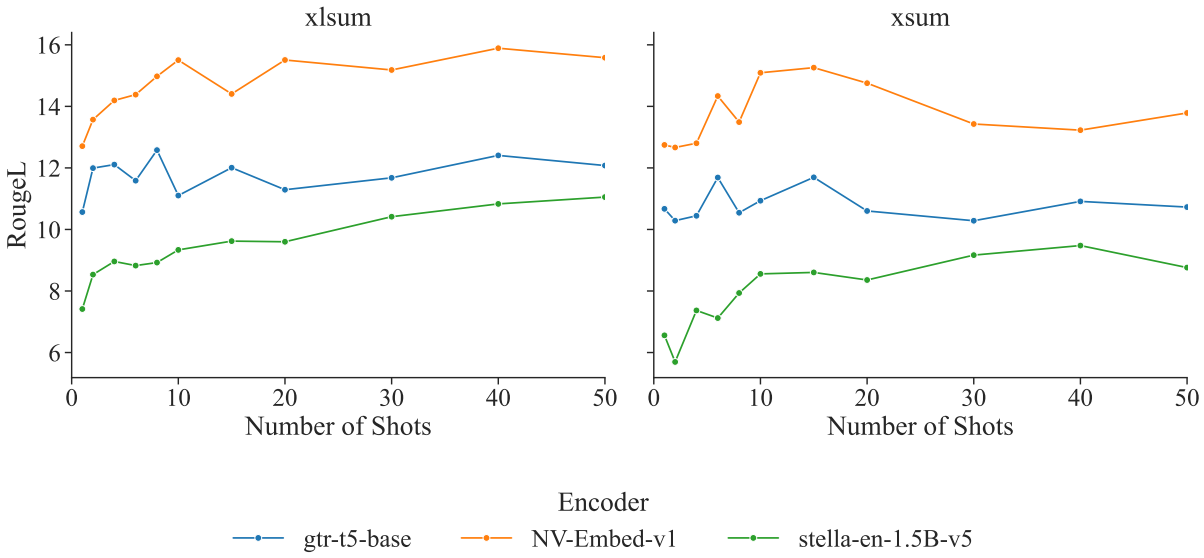


Figure B.6. Text Summarization - Pretrained Projectors

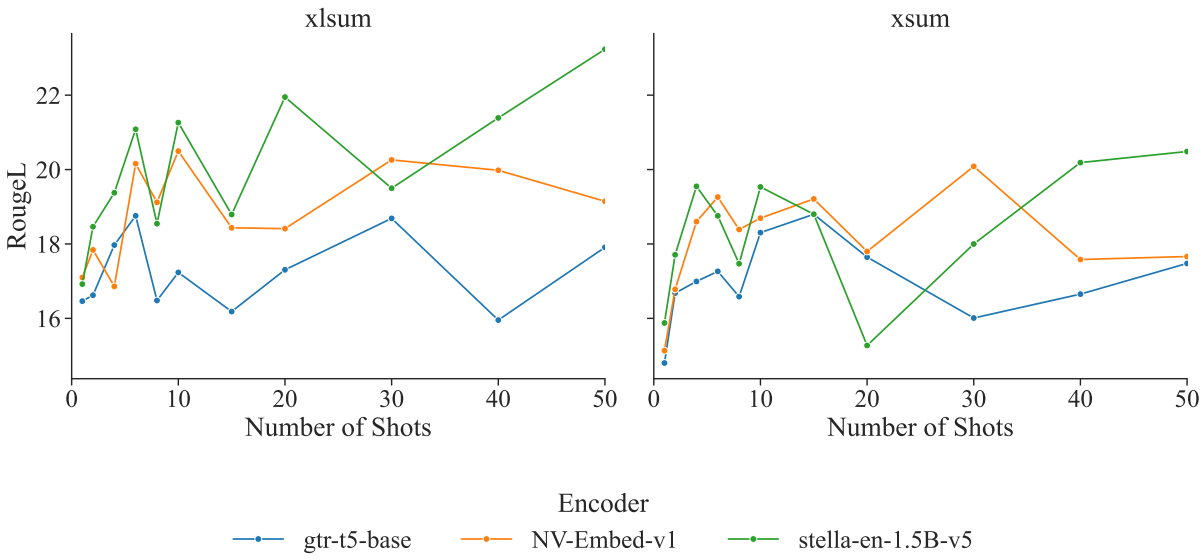


Figure B.7. Text Summarization - Finetuned Projectors

Table B.2. Question Categories and Their Associated Questions

Category	Questions
Sensory and Perceptual Descriptions	Does the input mention or describe a taste?
	Does the input mention or describe a sound?
	Does the sentence include a specific sound or auditory description?
	Does the input mention or describe a visual experience?
	Does the input mention or describe a texture?
	Does the sentence describe a sensory experience?
	Does the input mention anything related to color?
	Does the input mention or describe a smell?
	Does the input mention anything related to eyes?
	Does the sentence describe a visual experience or scene?
	Does the input describe a specific texture or sensation?
	Does the sentence describe a specific sensation or feeling?
Social, Emotional, and Interpersonal Content	Does the input mention anything related to arguing?
	Does the input mention anything related to empathy?
	Does the sentence involve a discussion about personal or social values?
	Does the input discuss a societal issue or social justice topic?
	Does the input mention or describe high emotional intensity?
	Does the sentence describe a relationship between people?
	Does the input mention or describe highly positive emotional valence?
	Does the input mention or describe highly negative emotional valence?
	Does the input mention anything related to conflict?
	Does the sentence describe a personal or social interaction that leads to a change or revelation?
	Does the sentence express a philosophical or existential query or observation?
	Does the sentence involve an expression of personal values or beliefs?
	Does the sentence express a sense of belonging or connection to a place or community?
	Is the sentence emotionally positive?
Cognitive and Reflective Aspects	Is the sentence reflective, involving self-analysis or introspection?
	Does the input involve planning or organizing?
	Does the text include a planning or decision-making process?
	Does the sentence convey a decision or choice made by the narrator?
	Does the sentence describe a personal reflection or thought?
	Is the input about a discovery or realization?
	Does the input contain a sense of ambiguity?
	Is the sentence providing an explanation or rationale?
	Does the input mention anything related to knowledge?
Language, Structure, and Syntax	Does the sentence contain a proper noun?
	Does the sentence include a conditional clause?
	Does the sentence contain a negation?
	Does the sentence use a unique or unusual word?
	Does the sentence include a direct speech quotation?
	Does the sentence include dialogue?
	Does the sentence contain a cultural reference?
	Does the sentence involve a recount of a social or community event?
	Does the input include a comparison or metaphor?
	Does the sentence include technical or specialized terminology?
	Is the sentence abstract rather than concrete?
	Does the sentence include an account of a miscommunication or misunderstanding?
	Does the text describe a mode of communication?
Physical Actions and Movements	Is the sentence conveying the narrator's physical movement or action in detail?
	Does the input mention anything related to motor movements?
	Does the sentence describe a physical action?
	Does the sentence describe a physical sensation?
	Does the sentence describe an activity related to daily life or routine?
	Does the input mention anything related to an action?
	Does the sentence involve spatial reasoning?
Numbers and Quantitative Information	Does the input mention a number less than 5?
	Does the input contain a number?
	Does the input mention a number greater than 100?
	Does the input mention anything related to arithmetic?
	Does the input mention anything related to calculation?
	Does the input contain a measurement?
Health, Disease, and Biological Aspects	Does the input mention anything related to diseases?
	Does the input mention anything related to food?
	Does the input mention anything related to age?
	Does the input mention anything related to gender?
	Does the input mention anything related to disgust?
	Does the input mention anything related to children?
Conflict, Urgency, and Change	Does the sentence involve an unexpected incident or accident?
	Does the input mention anything related to anger?
	Does the sentence convey a sense of urgency or haste?
	Does the sentence describe a change in a physical or emotional state?
	Does the sentence describe a moment of relief or resolution of tension?
	Does the sentence express the narrator's opinion or judgment about an event or character?

Appendix C

Supplementary Material for Chapter 3

C.1 Training Configurations

We employed the DeepSpeed [Rasley et al., 2020] framework for distributed training across four GPU nodes, each equipped with four AMD MI250 GPUs. We used vLLM [Kwon et al., 2023] for inference. Our implementation builds upon OpenRLHF [Hu et al., 2024] for both SFT and DPO. Given the input sequence length of up to 128K tokens, we leveraged FlashAttention-2 [Dao, 2023] alongside Ring Attention [Liu et al., 2023] to efficiently process extremely long sequences. The detailed hyperparameters for SFT and DPO are provided in Table C.1 and Table C.2.

Table C.1. Hyperparameters for SFT.

Hyperparameter	**Value**
Learning Rate	5e-7
Learning Rate Schedule	Cosine Annealing
Optimizer	Adam
β_1	0.9
β_2	0.95
Training dtype	bf16
Batch Size	128
Max Length	131,072

C.2 Short Context Performance

As shown in Table C.3, we evaluate the short-context perfromance across six tasks: ARC Easy, ARC Challenge, GSM8K, MathQA, MMLU, and MMLU Pro. AgenticLU performs on par with the base

Table C.2. Hyperparameters for DPO.

Hyperparameter	Value
Learning Rate	5e-7
Learning Rate Schedule	Cosine Annealing
Optimizer	Adam
β_1	0.9
β_2	0.95
Training dtype	bf16
Batch Size	128
β	0.1
Max Length	131,072

Table C.3. Performance comparison of AgenticLU and Llama3.1-8B-Instruct on short-context benchmarks. Scores represent accuracy percentages, with AgenticLU demonstrating matching results across tasks.

Model	ARC Easy	ARC Challenge	GSM8k	MathQA	MMLU	MMLU Pro	Avg
Llama3.1-8B	84.80	59.64	80.13	42.88	68.72	37.71	62.31
AgenticLU-8B	83.96	58.36	80.51	41.74	68.38	37.51	61.74

model Llama3.1-8B-Instruct on short-context benchmarks, demonstrating that AgenticLU preserves the original short-context ability while greatly enhancing long-context performance.

C.3 Agentic Workflow without Training

We conducted an additional experiment for prompting only with our agentic workflow, and we find that with an option to generate clarifications the model does get better on multi-hop QA questions (HotPotQA), but it is generally difficult for the base model to point to the correct paragraphs directly without any training, hence often resulting in same or slightly worse performance.

Results are listed in Table C.4, the context length is 128K tokens.

Table C.4. Performance comparison across different QA benchmarks with 128K token context length, with or without training.

Model	HotpotQA	NaturalQ	PopQA	TriviaQA	Avg
Llama-3.1-8B	40.0	56.1	56.1	80.6	58.2
Llama-3.1-8B + Prompting Only	53.3	56.7	51.6	72.8	58.6
AgenticLU-8B	71.1	77.8	65.5	88.3	75.7

Table C.5. Long-context performance on HotpotQA.

	HotpotQA				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	63.3	56.7	61.1	47.8	40.0
Llama3.1-8B+step-by-step	60.0	66.7	56.7	58.9	56.7
Llama3.1-8B+plan&solve	71.1	66.7	72.2	62.2	50.0
Llama3.1-8B+fact&reflect	58.9	58.9	62.2	61.1	48.9
ProLong-8B	62.2	65.6	57.8	53.3	58.9
Llama3.1-8B+LongRAG	61.1	58.9	73.3	56.7	57.8
AgenticLU-8B	81.1	75.6	78.9	75.6	71.1

Table C.6. Long-context performance on Nature Questions.

	Nature Questions				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	71.7	69.4	70.6	73.9	56.1
Llama3.1-8B+step-by-step	66.7	66.1	58.9	55.6	38.9
Llama3.1-8B+plan&solve	67.8	71.7	66.7	62.2	50.6
Llama3.1-8B+fact&reflect	63.3	63.3	61.7	59.4	40.0
ProLong-8B	83.3	82.2	83.9	90.0	77.8
Llama3.1-8B+LongRAG	65.6	76.1	79.4	77.2	73.9
AgenticLU-8B	91.7	91.1	85.0	85.0	77.8

C.4 Detailed Results on Seven Benchmark Tasks

As shown in Table C.5, Table C.6, Table C.7, Table C.8, Table C.9, Table C.10 and Table C.11, we evaluate the long-context performance across seven tasks: HotpotQA, Natural Questions, TriviaQA, PopQA, NarrativeQA, InfiniteBench QA and InfiniteBench Multiple-Choice. AgenticLU provides significant improvement for all tasks, especially for those that require multi-hop reasoning such as HotPotQA.

C.5 Evaluation Template

We use GPT-4o [OpenAI, 2023] to judge if the model's answer is correct. The specific prompt template with the structured output class is shown below.

Table C.7. Long-context performance on TriviaQA.

	TriviaQA				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	82.8	86.7	85.6	81.1	80.6
Llama3.1-8B+step-by-step	84.4	86.1	90.0	82.2	57.2
Llama3.1-8B+plan&solve	78.9	88.3	89.4	87.2	86.7
Llama3.1-8B+fact&reflect	87.8	83.9	84.4	86.7	84.4
ProLong-8B	71.1	88.3	78.9	82.8	78.3
Llama3.1-8B+LongRAG	77.2	79.4	83.9	83.9	83.3
AgenticLU-8B	88.3	92.2	91.1	93.3	88.3

Table C.8. Long-context performance on PopQA.

	PopQA				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	61.1	62.8	57.2	58.3	56.1
Llama3.1-8B+step-by-step	61.7	58.9	55.0	58.9	60.6
Llama3.1-8B+plan&solve	62.2	63.3	58.9	55.0	61.1
Llama3.1-8B+fact&reflect	65.0	64.4	58.9	53.3	65.0
ProLong-8B	67.8	68.3	70.0	64.4	65.6
Llama3.1-8B+LongRAG	47.8	54.4	54.4	57.2	50.6
AgenticLU-8B	82.2	82.2	78.3	76.7	65.6

Verification Prompts

```
Please verify the following answer:

Question: question Ground Truth Answers: ground truth Predicted Answer:

answer

Your task is to determine whether the predicted answer correctly matches

the ground truth. Focus on overall correctness and provide a detailed

explanation in the following format:

class VerificationResult:

explanation: str # Justification

confidence: float # Confidence score in the range [0,1]

correct_answer: bool # True if the prediction is correct, otherwise False
```

Table C.9. Long-context performance on NarrativeQA.

	NarrativeQA				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	15.0	19.0	27.0	35.0	38.0
Llama3.1-8B+step-by-step	23.0	30.0	36.0	51.0	43.0
Llama3.1-8B+plan&solve	22.0	25.0	38.0	41.0	39.0
Llama3.1-8B+fact&reflect	18.0	35.0	37.0	42.0	46.0
ProLong-8B	18.0	27.0	28.0	38.0	42.0
Llama3.1-8B+LongRAG	23.3	23.3	50.0	50.0	46.7
AgenticLU-8B	27.0	35.0	41.0	49.0	56.0

Table C.10. Long-context performance on InfbenchQA.

	InfbenchQA				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	17.0	31.0	36.0	40.0	48.0
Llama3.1-8B+step-by-step	21.0	36.0	36.0	45.0	43.0
Llama3.1-8B+plan&solve	17.0	26.0	32.0	41.0	40.0
Llama3.1-8B+fact&reflect	19.0	30.0	40.0	42.0	37.0
ProLong-8B	16.0	31.0	29.0	31.0	45.0
Llama3.1-8B+LongRAG	16.7	23.3	36.7	43.3	36.7
AgenticLU-8B	25.0	39.0	42.0	47.0	50.0

Table C.11. Long-context performance on InfbenchChoice.

	InfbenchChoice				
Model	8K	16K	32K	64K	128K
Llama3.1-8B	9.0	12.0	24.0	39.0	55.0
Llama3.1-8B+step-by-step	15.0	13.0	41.0	41.0	44.0
Llama3.1-8B+plan&solve	27.0	15.0	48.0	55.0	58.0
Llama3.1-8B+fact&reflect	20.0	14.0	38.0	51.0	56.0
ProLong-8B	22.0	27.0	37.0	48.0	58.0
Llama3.1-8B+LongRAG	16.7	30.0	43.3	53.3	63.3
AgenticLU-8B	45.0	46.0	47.0	64.0	68.0

C.6 Chain-of-Clarifications Workflow

The input was first processed into chunks and grouped with paragraph tags. We list the example prompts used in AgenticLU workflow below. In training, we sampled 100 variations of the same prompt text and use them randomly to avoid training collapse.

Chain-of-Clarifications Workflow Prompts

```
[System Prompt]
You are an AI assistant specialized in long context reasoning. Analyze
information thoroughly while maintaining clarity and focus. Track the
full context of conversations, building connections between concepts and
flagging when context review is needed. Break down complex problems into
components, showing your reasoning steps and stating key assumptions.
Structure your responses with clear headers and periodic summaries.
Present evidence for your conclusions, acknowledge uncertainties,
and request clarification when needed. Keep your analysis organized,
explicit, and focused on addressing the core question.
[Long-Context Input]
<para 1> [chunk 1] </para 1> <para 2> [chunk 2] </para 2> ... {Question}
[Self Clarification - Raise Question]
In order to answer this question, ask one question about what you want to
know in order to better answer it.
[Contextual Grounding - Pointback]
Help me find relevant context to answer the previous clarifying question.
[Self Clarification - Answer Question]
Based on the relevant context, answer the previous clarifying question.
[Answer the Original Question]
Now, let's answer the final question. Be concise in your answer.
```

Bibliography

Abhineet Agarwal, Yan Shuo Tan, Omer Ronen, Chandan Singh, and Bin Yu. Hierarchical shrinkage: improving the accuracy and interpretability of tree-based methods. *arXiv:2202.00858*, 2 2022. arXiv: 2202.00858.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, et al. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*, 2024.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36, 2024.

Vinamra Benara, Chandan Singh, John X Morris, Richard Antonello, Ion Stoica, Alexander G Huth, and Jianfeng Gao. Crafting interpretable embeddings by asking llms questions. *arXiv preprint arXiv:2405.16714*, 2024.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*,

2024.

Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2:3, 2023.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and llms by learning to learn discrete functions. *arXiv preprint arXiv:2310.03016*, 2023.

Gérard Biau, Erwan Scornet, and Johannes Welbl. Neural random forests. *Sankhya A*, 81(2):347–386, 2019.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 10 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

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*, 33:1877–1901, 2020.

Miguel A Carreira-Perpinán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in neural information processing systems*, 31, 2018.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. Parallel structures in pre-training data yield in-context learning. *arXiv preprint arXiv:2402.12530*, 2024.

Hugh Chipman and Robert E McCulloch. Hierarchical priors for bayesian cart shrinkage. *Statistics and Computing*, 10:17–24, 2000.

Hugh A Chipman, Edward I George, and Robert E McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind

Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, 2023.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023.

Carl Edwards, Qingyun Wang, Lawrence Zhou, and Heng Ji. L+m-24: Building a dataset for language+molecules @ acl 2024. *arXiv preprint arXiv:2403.00791*, 2024.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930): 47–53, 2022.

Benjamin Feuer, Chinmay Hegde, and Niv Cohen. Scaling tabpfn: Sketching and feature selection for tabular prior-data fitted networks. *arXiv preprint arXiv:2311.10609*, 2023.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv preprint arXiv:2310.17086*, 2023.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). *arXiv preprint arXiv:2410.02660*, 2024.

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

Angeliki Giannou, Liu Yang, Tianhao Wang, Dimitris Papailiopoulos, and Jason D Lee. How well can transformers emulate in-context newton's method? *arXiv preprint arXiv:2403.03183*, 2024.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Unlocking the power of retrieval-augmented tabular deep learning. *arXiv preprint arXiv:2307.14338*, 2023.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

Etienne Grossmann. Adatree: boosting a weak classifier into a decision tree. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 105–105. IEEE, 2004.

Suryabhan Singh Hada, Miguel Á Carreira-Perpiñán, and Arman Zharmagambetov. Sparse oblique decision trees: A tool to understand and manipulate neural net features. *Data Mining and Knowledge Discovery*, pages 1–40, 2023.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *arXiv preprint arXiv:2305.00586*, 2023.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online, August 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.findings-acl.413.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Munir Hiabu, Enno Mammen, and Joseph T Meyer. Random planted forest: a directly interpretable tree ensemble. *arXiv preprint arXiv:2012.14563*, 2020.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Jian Hu, Xibin Wu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Yixing Jiang, Jeremy Irvin, Ji Hun Wang, Muhammad Ahmed Chaudhry, Jonathan H Chen, and Andrew Y Ng. Many-shot in-context learning in multimodal foundation models. *arXiv preprint arXiv:2405.09798*, 2024.

Nathanael Jo, Sina Aghaei, Jack Benson, Andres Gomez, and Phebe Vayanos. Learning optimal fair decision trees: Trade-offs between interpretability, fairness, and accuracy. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 181–192, 2023.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Aaron E Kornblith, Chandan Singh, Gabriel Devlin, Newton Addo, Christian J Streck, James F Holmes, Nathan Kuppermann, Jacqueline Grupp-Phelan, Jeffrey Fineman, Atul J Butte, and Bin Yu. Predictability and stability testing to assess clinical decision instrument performance for children after blunt torso trauma. *medRxiv*, 2022. doi: 10.1101/2022.03.08.22270944. URL https://www.medrxiv.org/content/early/2022/03/08/2022.03.08.22270944.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 453–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*, 2022.

Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

Amanda LeBel, Lauren Wagner, Shailee Jain, Aneesh Adhikari-Desai, Bhavin Gupta, Allyson Morgenthal, Jerry Tang, Lixiang Xu, and Alexander G Huth. A natural language fmri dataset for voxelwise encoding models. *bioRxiv*, pages 2022–09, 2022.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2024.

Guang-He Lee and Tommi S Jaakkola. Oblique decision trees from derivatives of relu networks. *arXiv preprint arXiv:1909.13488*, 2019.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*, 2023a.

Mukai Li, Shansan Gong, Jiangtao Feng, Yiheng Xu, Jun Zhang, Zhiyong Wu, and Lingpeng Kong. In-context learning with many demonstration examples. *arXiv preprint arXiv:2302.04931*, 2023b.

Siheng Li, Cheng Yang, Zesen Cheng, Lemao Liu, Mo Yu, Yujiu Yang, and Wai Lam. Large language models can self-improve in long-context reasoning. *arXiv preprint arXiv:2411.08147*, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pages 19565–19594. PMLR, 2023c.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

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 arXiv:2307.03576*, 2023.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*, 2022.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic

orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

Hariharan Manikandan, Yiding Jiang, and J Zico Kolter. Language models are weak learners. *arXiv preprint arXiv:2306.14101*, 2023.

Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfrembedding-mistral: enhance text retrieval with transfer learning. *Salesforce AI Research Blog*, 3, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, 2022a.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022b.

John X Morris, Chandan Singh, Alexander M Rush, Jianfeng Gao, and Yuntian Deng. Tree prompting: efficient task adaptation without fine-tuning. *arXiv preprint arXiv:2310.14034*, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2021.

W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*, 116(44):22071–22080, 2019. doi: 10.1073/pnas.1900654116.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International Conference on Machine Learning*, pages 16569–16594. PMLR, 2022.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*, 2021.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

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.

Soma Onishi, Kenta Oono, and Kohei Hayashi. Tabret: Pre-training transformer-based tabular models for unseen columns. *arXiv preprint arXiv:2303.15747*, 2023.

OpenAI. GPT-4 technical report, 2023.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Fabian Pedregosa and Philippe Gervais. memory-profiler: A module for monitoring memory usage of a python program. https://pypi.org/project/memory-profiler/, 2021.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlock-

ing multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Joseph D Romano, Trang T Le, William La Cava, John T Gregg, Daniel J Goldberg, Praneel Chakraborty, Natasha L Ray, Daniel Himmelstein, Weixuan Fu, and Jason H Moore. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2*, 2021.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*, 2018.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL https://www.aclweb.org/anthology/D18-1404.

Jingbo Shang, Zai Zheng, Jiale Wei, Xiang Ying, Felix Tao, and Mindverse Team. Ai-native memory: A pathway from llms towards agi. *arXiv preprint arXiv:2406.18312*, 2024.

Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. imodels: a python package for fitting interpretable models. *Journal of Open Source Software*, 6(61):3192, 2021.

Chandan Singh, Armin Askari, Rich Caruana, and Jianfeng Gao. Augmenting interpretable models with large language models during training. *Nature Communications*, 14(1):7913, 2023.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Daria Sorokina, Rich Caruana, and Mirek Riedewald. Additive groves of regression trees. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*, pages 323–334. Springer, 2007.

Yan Shuo Tan, Chandan Singh, Keyan Nasseri, Abhineet Agarwal, and Bin Yu. Fast interpretable greedy-tree sums (figs). *arXiv:2201.11931 [cs, stat]*, 1 2022. arXiv: 2201.11931.

Jerry Tang, Amanda LeBel, Shailee Jain, and Alexander G Huth. Semantic reconstruction of continuous language from non-invasive brain recordings. *Nature Neuroscience*, 26(5):858–866, 2023.

Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pages 6166–6175. PMLR, 2019.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218. Citeseer, 2012.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Gilmer Valdes, José Marcio Luna, Eric Eaton, Charles B Simone, Lyle H Ungar, and Timothy D Solberg. Mediboost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Scientific reports*, 6(1):37854, 2016.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL http://doi.acm.org/10.1145/2641190.264119.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren,

Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024a.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658*, 2024b.

Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Jingfeng Wu, Difan Zou, Zixiang Chen, Vladimir Braverman, Quanquan Gu, and Peter Bartlett. How many pretraining tasks are needed for in-context learning of linear regression? In *The Twelfth International Conference on Learning Representations*, 2023.

Junda Wu, Zhehao Zhang, Yu Xia, Xintong Li, Zhaoyang Xia, Aaron Chang, Tong Yu, Sungchul Kim, Ryan A Rossi, Ruiyi Zhang, et al. Visual prompting in multimodal large language models: A survey. *arXiv preprint arXiv:2409.15310*, 2024.

Steve Yadlowsky, Lyric Doshi, and Nilesh Tripuraneni. Can transformer models generalize via in-context learning beyond pretraining data? In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*, 2024.

Olcay Taner Yıldız and Ethem Alpaydın. Regularizing soft decision trees. In *Information Sciences and Systems 2013: Proceedings of the 28th International Symposium on Computer and Information Sciences*, pages 15–21. Springer, 2013.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Valentina Zantedeschi, Matt Kusner, and Vlad Niculae. Learning binary decision trees by argmin differentiation. In *International Conference on Machine Learning*, pages 12298–12309. PMLR, 2021.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Dun Zhang. Stella, 2024. URL https://huggingface.co/dunzhang/stella_en_1.5B_v5.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814/.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arik. Chain of agents: Large language models collaborating on long-context tasks. *arXiv preprint arXiv:2406.02818*, 2024b.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. LongRAG: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.

emnlp-main.1259. URL https://aclanthology.org/2024.emnlp-main.1259/.

Siyan Zhao, John Dang, and Aditya Grover. Group preference optimization: Few-shot alignment of large language models. *arXiv preprint arXiv:2310.11523*, 2023.

Siyan Zhao, Tung Nguyen, and Aditya Grover. Probing the decision boundaries of in-context learning in large language models. *arXiv preprint arXiv:2406.11233*, 2024b.

Xiaowei Zhao, Yong Zhou, and Xiujuan Xu. Dual encoder: Exploiting the potential of syntactic and semantic for aspect sentiment triplet extraction. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5401–5413, Torino, Italia, May 2024c. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.480/.

Xinran Zhao, Hongming Zhang, Xiaoman Pan, Wenlin Yao, Dong Yu, Tongshuang Wu, and Jianshu Chen. Fact-and-reflection (FaR) improves confidence calibration of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8702–8718, Bangkok, Thailand, August 2024d. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.515. URL https://aclanthology.org/2024.findings-acl.515/.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? A study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33: 18330–18341, 2020.

Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Learning a decision tree algorithm with transformers. *arXiv preprint arXiv:2402.03774*, 2024a.

Yufan Zhuang, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. Vector-icl: In-context learning with continuous vector representations. *arXiv preprint arXiv:2410.05629*, 2024b.

Yufan Zhuang, Xiaodong Yu, Jialian Wu, Ximeng Sun, Ze Wang, Jiang Liu, Yusheng Su, Jingbo Shang, Zicheng Liu, and Emad Barsoum. Self-taught agentic long context understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5525–5537, 2025.