Extrapolation by Association: Length Generalization Transfer in Transformers

Anonymous Authors[1]

Abstract

Transformer language models have demonstrated impressive generalization capabilities in natural language domains, yet we lack a fine-grained understanding of how such generalization arises. In this paper, we investigate length generalization—the ability to extrapolate from shorter to longer inputs—through the lens of *task association*. We find that length generalization can be *transferred* across related tasks. That is, training a model with a longer and related auxiliary task can lead it to generalize to unseen and longer inputs from some other target task. We demonstrate this length generalization transfer across diverse algorithmic tasks, including arithmetic operations, string transformations, and maze navigation. Our results show that transformer models can inherit generalization capabilities from similar tasks when trained jointly. Moreover, we observe similar transfer effects in pretrained language models, suggesting that pretraining equips models with reusable computational scaffolding that facilitates extrapolation in downstream settings. Finally, we provide initial mechanistic evidence that length generalization transfer correlates with the re-use of the same attention heads between the tasks. Together, our findings deepen our understanding of how transformers generalize to out-of-distribution inputs and highlight the compositional reuse of inductive structure across tasks.

1. Introduction

A central theme of transformer language models is their ability to generalize. By scaling up data and model size, large language models develop emergent abilities that exceed expectations (Wei et al., 2022). They can also transfer knowledge across domains and tasks (OpenAI, 2024; Brown et al., 2020; Sanh et al., 2022). While it is widely believed that language models are not simply parroting or memorizing their training data, we still lack a fine-grained understanding of how language models apply skills learned during training to potentially unseen problems.

The out-of-distribution (OOD) generalization capabilities of language models have garnered much attention in the literature (Anil et al., 2022; Zhang et al., 2024; Yang et al., 2024). In this work, we study a canonical example of OOD generalization, *length generalization*, which is the ability to generalize from shorter to longer inputs (Zhou et al., 2023). There is a long line of work focusing on improving length generalization of arithmetic tasks in transformers, which has spurred innovations in positional encoding schemes and transformer architecture (Cho et al., 2024; McLeish et al., 2024). Closely related is the concept of compositional generalization, where the model combines previously learned skills to solve new problems (Yang et al., 2024; Xu et al., 2024).

In this work, we study a new mechanism underlying length generalization: *extrapolation by association*. We hypothesize that, when faced with a problem outside its training distribution, language models can use related skills to solve it. Specifically, we ask: Can generalization to longer inputs in one task *transfer* to another task that is only trained on short examples?

To showcase the length generalization transfer capabilities in transformers, we choose three distinct groups of synthetic tasks. The tasks in each group are related such that they represent similar algorithmic procedures. Within each group, we train multiple tasks together, and crucially, we train an "auxiliary task" at a longer length and a "main task" at a shorter length. Using this setup, we observe
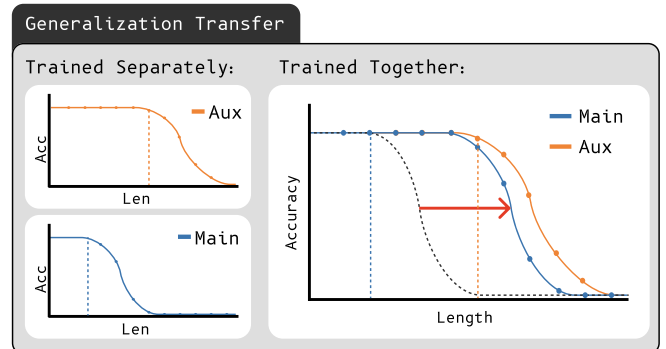

Figure 1: Trained separately, each task fails to generalize to longer inputs. When trained jointly, the main task inherits the generalization range of the auxiliary task.

1

that the shorter main task generalizes to the length of the longer auxiliary task when trained together. See Section 2 for the tasks and respective lengths used in each experiment.

Contributions

1. We present the phenomenon of **length generalization transfer**, in which transformer models trained on related tasks exhibit extrapolation behavior not present when trained on the target task alone, providing new insights on the effect of multitask training on length generalization.

2. We show that the same phenomenon exists in pretrained language models, and that natural language pretraining transfers length generalization capabilities to synthetic downstream tasks.

2. Experimental Settings

Models.   For from-scratch experiments, we use transformer models with 6 heads and 6 layers, following the Llama architecture (AI@Meta, 2024), which uses Rotary Positional Embeddings (RoPE) (Su et al., 2023) for position encoding. For experiments with pretrained models, we use SmolLM (Allal et al., 2024), which provides access to intermediate checkpoints during pretraining, allowing us to investigate how length generalization transfer evolves over time.

Tasks.   We evaluate length generalization transfer across three categories of algorithmic problems: arithmetic, string manipulation, and maze solving. Our tasks include:

- **Arithmetic Tasks**

  - `reverse add` – Compute the sum of two integers, with least significant digit first.
  - `no carry` – Compute digit-wise sums modulo 10, without carry propagation.
  - `carry only` – Output a binary mask indicating carry positions during addition.
  - `reverse subtract` – Compute the digit-wise difference between two numbers, with least significant digit first.
  - $n \times 3$ `CoT multiply` – Multiply an n-digit number by 3, with chain-of-thought steps.

- **String Manipulation Tasks**

  - `copy` – Return the input string unchanged.
  - `MQAR` (Multi-Query Associative Recall) (Arora et al., 2023) – Given a repeated query substring, retrieve the next character following each occurrence.
  - `flipcase` – Flip the case of all alphabetic characters (lower $\leftrightarrow$ upper).
  - `reverse` – Reverse the character order of the input string.
  - `flipcase-reverse` – Apply both reversal and case-flipping to the input string.

- **Maze Tasks** (Appendix A)

  - `DFS trace` – Simulate a depth-first search from a start node to a goal node in a maze.
  - `shortest path` – Return the optimal (shortest) path between a start and goal node.

Data sampling and Task Length.   Since we train under a multi-task setting, at each iteration, a task is sampled uniformly at random from a predefined task group. For the selected task, an individual training example is constructed based on a single governing parameter: *length*, which determines the size or complexity of the problem instance. The length of each example is sampled uniformly from a specified range for that task. All training data is generated on-the-fly during training.

Since the notion of *length* varies across task types, we define length for each task as:

- **Addition Tasks**: the maximum number of digits in both operands.

- **String Tasks**: the number of characters in the input string.

- **Maze Tasks**: the number of nodes in the input maze graph. See Appendix A for further details.

Task Groups. We construct *task groups* by pairing a main task, trained on short sequence lengths, with one or more auxiliary tasks, trained on longer sequences. The main goal is to evaluate whether training on a related auxiliary task improves the main task's ability to generalize to longer inputs, despite never seeing such lengths during training. The list of task groups are:

Main Task (Train Length)	Auxiliary Task(s) (Train Length)
`reverse add` (16)	`no carry` & `carry only` (32)
`reverse add` (16)	`reverse subtract` (32)
`reverse add` (8)	$n \times 3$ `CoT multiply` (16)
`string copy` (16)	`MQAR` (32)
`capitalize-reverse` (16)	`capitalize` (32), `reverse` (32)
`DFS trace` (32)	`shortest path` (64)

Training and Evaluation. Each example consists of an input-output pair. We use a loss mask to train only on output tokens (and for MQAR, only on answer characters). At test time, we evaluate using exact match accuracy on a fixed test set of 1024 examples. For each configuration, we report results across 5 random initialization seeds but the dataset is kept the same. Full experimental configurations and hyperparameter details are provided in Appendix E.

3. Length Generalization Transfer in Algorithmic Tasks

In this section, we demonstrate that while length generalization is often difficult for algorithmic tasks, it can emerge through transfer when the model is co-trained on longer auxiliary tasks. We present results from arithmetic and maze navigation tasks, with additional analyses of string manipulation tasks available in Appendix 3.2.

3.1. Arithmetic Tasks

Reverse addition has become a popular synthetic task for studying length generalization (Lee et al., 2023; Shen et al., 2023; Zhou et al., 2023; 2024; Cho et al., 2024; McLeish et al., 2024; Lee et al., 2025) in Transformers. The task involves calculating the sum of two randomly sampled integers, and length generalization in this task involves training on examples up to some fixed length, and generalizing on test data beyond the training lengths. Here, we adopt the `reverse add` format proposed by (Lee et al., 2023), where the operands and the sum are reversed for faster learning. For the auxiliary tasks, we consider (1) `reverse subtract`, which computes the difference between two operands, (2) `no carry`, which computes the digit-wise sum mod 10, ignoring the carries, and (3) `carry only`, which computes the locations where a carry happens in the addition.

As shown in Figure 2, models trained only on `reverse add` (Figure 2d) struggle to generalize beyond the training length. However, when co-trained with longer auxiliary tasks (Figures 2a, 2b, 2c), the model successfully extrapolates, often matching the auxiliary task's generalization range. This provides empirical evidence that length generalization can transfer across tasks. It is worth noting that the generalization behavior is not entirely robust: different random seeds yield noticeably different outcomes, suggesting unstable training dynamics. We discuss this instability further in Appendix C.4.

3.2. String Manipulation Tasks

We observe transfer effects similar to arithmetic tasks on two string task groups. The tasks include: `string copy`, which returns the input unchanged; `MQAR` (Multi-Query Associative Recall) (Arora et al., 2023), where the model retrieves the next character given a random substring; `reverse`, which reverses character order; `capitalize`, which inverts letter case; and `capitalize-reverse`, combining case inversion and reversal.

Figure 3 shows that when trained on main tasks alone (Figures 3b, 3d), the model does not generalize beyond the training range. On the other hand, training with auxiliary tasks enables substantial extrapolation (as shown in Figures 3a and 3c).

3.3. Maze Tasks

We further present results on the maze task group `Shortest Path` and `DFS Trace` in Appendix A.

(a) Aux: `no carry` & `carry only`

(b) Aux: `reverse subtract`

(c) Aux: $n \times 3$ `CoT multiply`
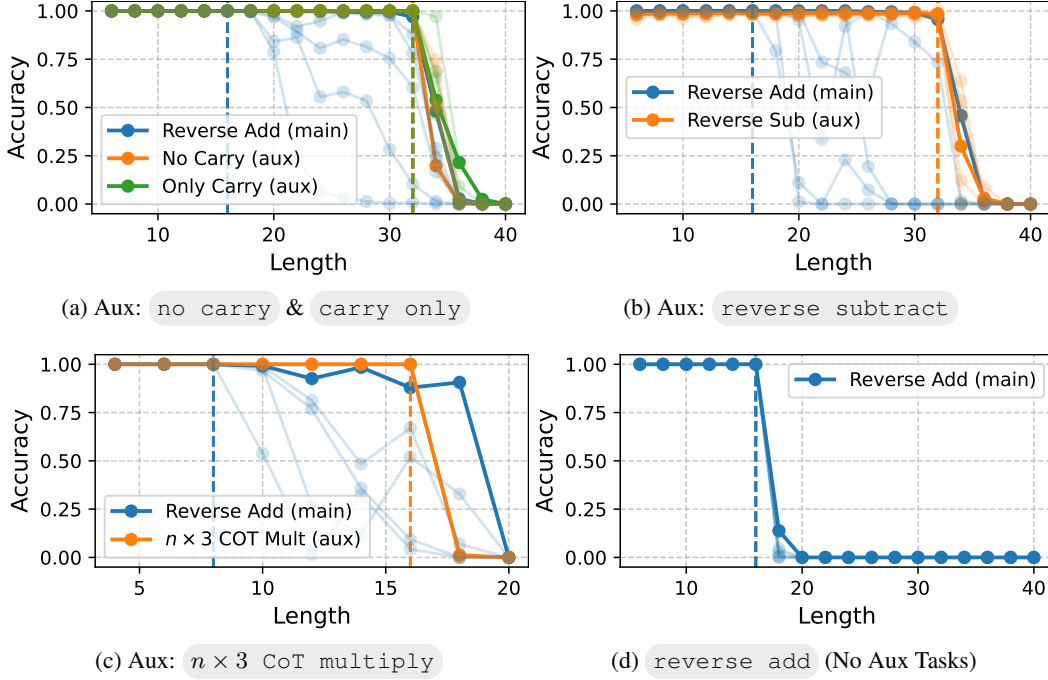
(d) `reverse add` (No Aux Tasks)

Figure 2: Length generalization results for addition-related task groups. The main task is `reverse add`, with performance shown when trained with different auxiliary tasks. Each model is trained with 5 random seeds; best-performing runs are shown in bold. The dashed vertical line indicates the maximum training length for each task. When trained alone (d), the model fails to generalize beyond training length. Co-training with related auxiliary tasks (a-c) enables extrapolation to longer inputs.

(a) Main: `string copy`, Aux: `MQAR`

(b) `string copy` (No Aux Tasks)

(c) Aux: `capitalize` & `reverse`
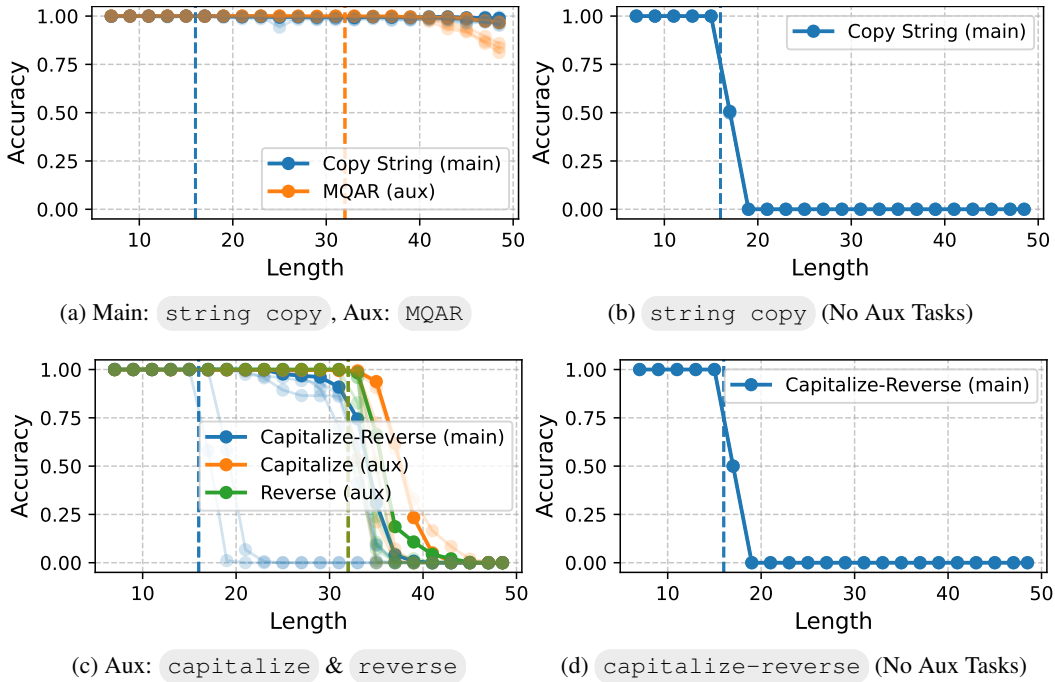
(d) `capitalize-reverse` (No Aux Tasks)

Figure 3: Performance plots for string tasks. When trained alone (b, d), models fail to generalize beyond their training range. Co-training with auxiliary tasks (a, c) enables substantial length extrapolation.

4

References

Amirhesam Abedsoltan, Huaqing Zhang, Kaiyue Wen, Hongzhou Lin, Jingzhao Zhang, and Mikhail Belkin. Task generalization with autoregressive compositional structure: Can learning from d tasks generalize to d^t tasks? *arXiv preprint arXiv:2502.08991*, 2025.

Kartik Ahuja and Amin Mansouri. On provable length and compositional generalization, 2024. URL https://arxiv.org/abs/2402.04875.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. Smollm - blazingly fast and remarkably powerful, 2024.

Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=zSkYVeX7bC4.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models, 2023. URL https://arxiv.org/abs/2312.04927.

Pranjal Awasthi and Anupam Gupta. Improving length-generalization in transformers via task hinting, 2023. URL https://arxiv.org/abs/2310.00726.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Velivckovi'c. Round and round we go! what makes rotary positional encodings useful? *ArXiv*, abs/2410.06205, 2024.

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

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. doi: 10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. Position coupling: Improving length generalization of arithmetic transformers using task structure. 2024. URL https://api.semanticscholar.org/CorpusID:273695226.

Yiran Ding, L. Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending llm context window beyond 2 million tokens. *ArXiv*, abs/2402.13753, 2024.

Shaoxiong Duan, Yining Shi, and Wei Xu. From interpolation to extrapolation: Complete length generalization for arithmetic transformers. *arXiv preprint arXiv:2310.11984*, 2023.

Yann Dubois, Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. Location attention for extrapolation to longer sequences. *arXiv preprint arXiv:1911.03872*, 2019.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

Arian Hosseini, Alessandro Sordoni, Daniel Toyama, Aaron Courville, and Rishabh Agarwal. Not all llm reasoners are created equal. *arXiv preprint arXiv:2410.01748*, 2024.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges, 2025. URL https://arxiv.org/abs/2502.01612.

Mingchen Li, Xuechen Zhang, Yixiao Huang, and Samet Oymak. On the power of convolution-augmented transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18393–18402, 2025.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*, 2023.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *arXiv preprint arXiv:2405.17399*, 2024.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D Manning. The eos decision and length extrapolation. *arXiv preprint arXiv:2010.07174*, 2020.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022. URL https://arxiv.org/abs/2209.11895.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *ArXiv*, abs/2309.00071, 2023.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Philip Quirke and Fazl Barez. Understanding addition in transformers. *arXiv preprint arXiv:2310.13121*, 2023.

Philip Quirke, Clement Neo, and Fazl Barez. Arithmetic in transformers explained, 2025. URL https://arxiv.org/abs/2402.02619.

Rahul Ramesh, Ekdeep Singh Lubana, Mikail Khona, Robert P Dick, and Hidenori Tanaka. Compositional capabilities of autoregressive transformers: A study on synthetic, interpretable tasks. In *Forty-first International Conference on Machine Learning*.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*, 2023.

Mahdi Sabbaghi, George Pappas, Hamed Hassani, and Surbhi Goel. Explicitly encoding structural symmetry is key to length generalization in arithmetic tasks. *arXiv preprint arXiv:2406.01895*, 2024.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *ArXiv*, abs/2206.07682, 2022.

David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 296–303, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237880. URL https://doi.org/10.1145/237814.237880.

Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability, 2024. URL https://arxiv.org/abs/2407.15720.

Haoran Yang, Hongyuan Lu, Wai Lam, and Deng Cai. Exploring compositional generalization of large language models. In Yang (Trista) Cao, Isabel Papadimitriou, Anaelia Ovalle, Marcos Zampieri, Francis Ferraro, and Swabha Swayamdipta, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 16–24, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-srw.3. URL https://aclanthology.org/2024.naacl-srw.3/.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-mix: a flexible and expandable family of evaluations for ai models, 2023. URL https://arxiv.org/abs/2310.17567.

Xingxuan Zhang, Jiansheng Li, Wenjing Chu, Junjia Hai, Renzhe Xu, Yuqing Yang, Shikai Guan, Jiazheng Xu, and Peng Cui. On the out-of-distribution generalization of multimodal large language models, 2024. URL https://arxiv.org/abs/2402.06599.

Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Can models learn skill composition from examples?, 2025. URL https://arxiv.org/abs/2409.19808.

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.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

A. Length Generalization Transfer in Maze Tasks

We examine maze-solving tasks as a testbed for length generalization transfer. In our maze experiments, we define a maze as a spanning tree over a square grid, generated using Wilson's algorithm (Wilson, 1996), which ensures uniform sampling via loop-erased random walks. For each problem instance, we randomly sample a start and end node, and the model is tasked with producing a path from start to end. Maze inputs are represented as adjacency lists, with each node and its neighbors encoded as individual tokens (e.g., [1], [2], ..., [64]). The output solution path is encoded as a list of nodes in the same format.

A key challenge in defining length generalization for maze solving is that increasing grid size would introduce unseen node tokens at test time. To avoid this issue, we fix the underlying grid size and instead define length as the *number of nodes* included in the maze graph. We generate partial mazes by running Wilson's algorithm until the desired number of nodes is connected, creating valid traversal problems that don't cover the full grid. For example, to construct a 32-node maze on an 8×8 grid, we stop the generation process once 32 nodes have been added to the spanning tree. Figure 4 illustrates such partial mazes with 16, 32, and 64 nodes, and Figure 5 shows the detailed input/output formatting.
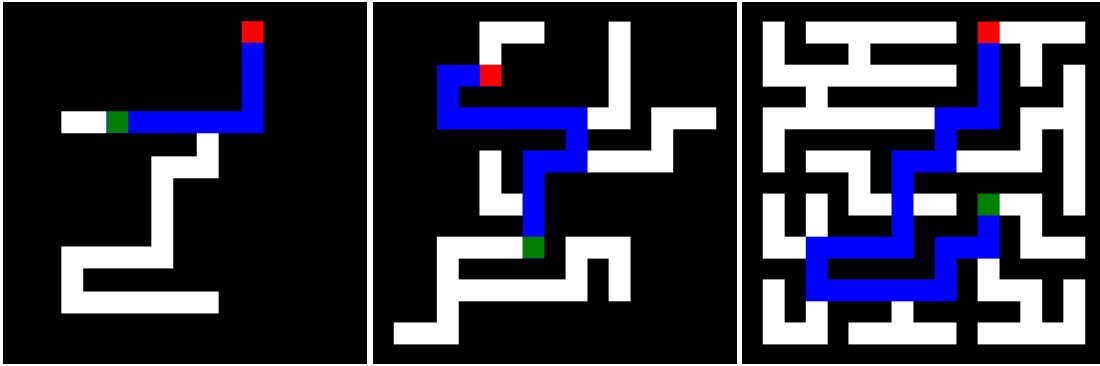

Figure 4: 8×8 mazes with number of nodes equal to 16, 32, and 64. We define length generalization as the ability to generalize to mazes with a higher number of nodes.

We consider two maze tasks: (1) `shortest path`, where the model outputs the shortest path from start to end node, and (2) `DFS trace`, where the model simulates a depth-first search traversal (including backtracking). Shortest path is harder to learn perfectly, as it requires "lookahead" at branch points, while DFS trace allows exploration and backtracking.

A.1. Maze Results

Figure 6 shows that in the multi-task setting, the addition of `shortest path` helps `DFS trace` generalize to higher lengths. The opposite is true as well: `DFS trace` helps `shortest path` generalize to higher lengths, as shown in Section A.1.1.

A.1.1. TRANSFER WITH SWAPPED MAIN AND AUXILIARY TASKS

We consider another maze task group where we the main and auxiliary tasks are reversed relative to Section A. In this case, the main task is `shortest path`, and the auxiliary task is `DFS trace`. As shown in Figure 7, co-training with the auxiliary task again improves length generalization performance. While `shortest path` is more difficult than `DFS trace`, the model benefits from learning a related traversal strategy.

B. Related Works

Length Generalization. Length generalization concerns extrapolating to longer sequence lengths than those seen during training (Dubois et al., 2019; Hupkes et al., 2020; Newman et al., 2020; Anil et al., 2022). Previous approaches include architectural modifications such as specialized positional embeddings (Press et al., 2021; Li et al., 2023; Ruoss et al., 2023; Kazemnejad et al., 2024; Sabbaghi et al., 2024; Cho et al., 2024; Zhou et al., 2024; McLeish et al., 2024), looping (Fan et al., 2024), novel attention mechanisms (Duan et al., 2023; Li et al., 2025), and input format augmentation (Zhou et al.,
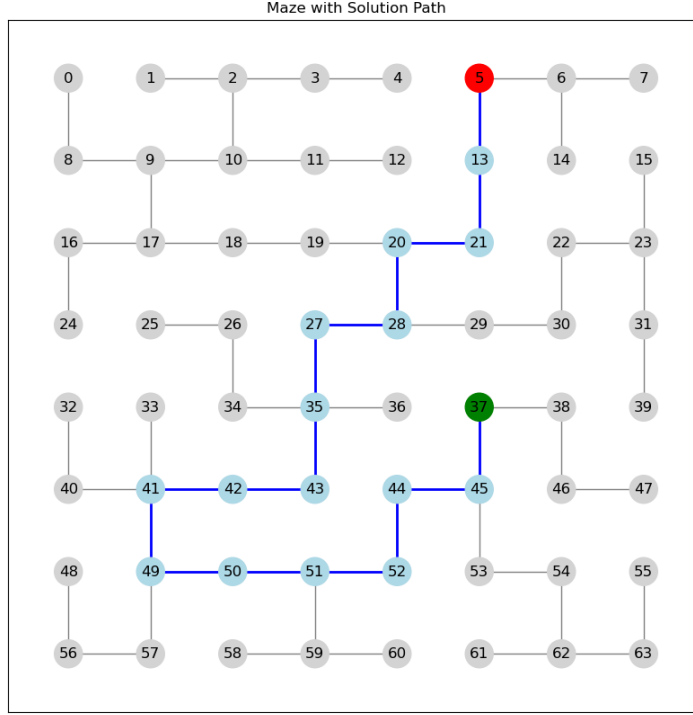
Figure 5: Detailed example of maze data format. Each node is a random number selected from $n \times n$ nodes in the grid.

2023; 2024). Beyond arithmetic, Yehudai et al. (2021) studies length generalization in graph tasks. In contrast, our work examines a novel mechanism from which length generalization emerges: transfer from related tasks. Finally, closely related to our work, "task hinting" (Awasthi and Gupta, 2023) trains sorting and increment-by-one tasks with simpler auxiliary tasks, showing improvements in length generalization performance.

Compositional Capabilities. To explain emergent capabilities in language models, many works study compositional generalization to understand whether transformers can gain abilities beyond those in the training set. Yu et al. (2023), Zhao et al. (2025) and Hosseini et al. (2024) design benchmarks testing the ability to combine learned skills to solve compositional math problems. Ahuja and Mansouri (2024) derive provable guarantees for length and compositional generalization conditioned on training set diversity. Some works use synthetic tasks to probe compositional generalization. (Ramesh et al.) show transformers achieve compositional generalization on unseen combinations using a series of bijections and permutations applied to strings, while (Abedsoltan et al., 2025) show similar results on families of parity functions.

For the specific task of addition, works like Quirke and Barez (2023) and Quirke et al. (2025) identify computational circuits responsible for compositional subtasks and show transferability of such circuits to the related task of subtraction.

C. Additional Results

C.1. Task Overview

See Figure 8

C.2. Control Tasks

To verify that length generalization transfer does not arise from merely seeing longer inputs, we further test arithmetic tasks and string operations with control auxiliary tasks. For arithmetic, we use `copy-first-op`, which follows the addition format but simply copies the first operand. For string operations, we pair `copy` with `reverse`. As expected, length generalization transfer is not observed with unrelated task (Figure 9).

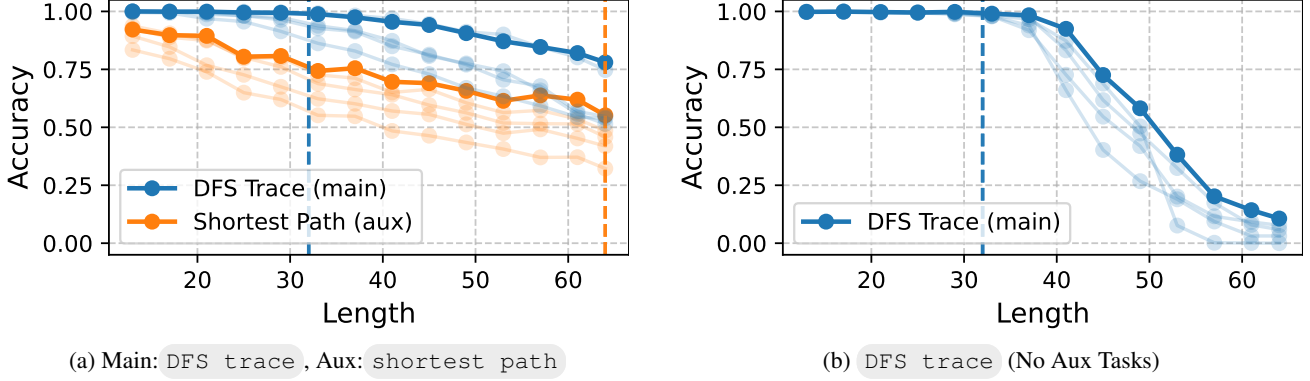(a) Main: `DFS trace`, Aux: `shortest path`

(b) `DFS trace` (No Aux Tasks)

Figure 6: Performance plots for maze tasks. Each model is trained with 5 random seeds; best-performing runs are shown in bold lines. Co-training `DFS trace` with `shortest path` (a) enables generalization to longer lengths compared to training on `DFS trace` alone (b).
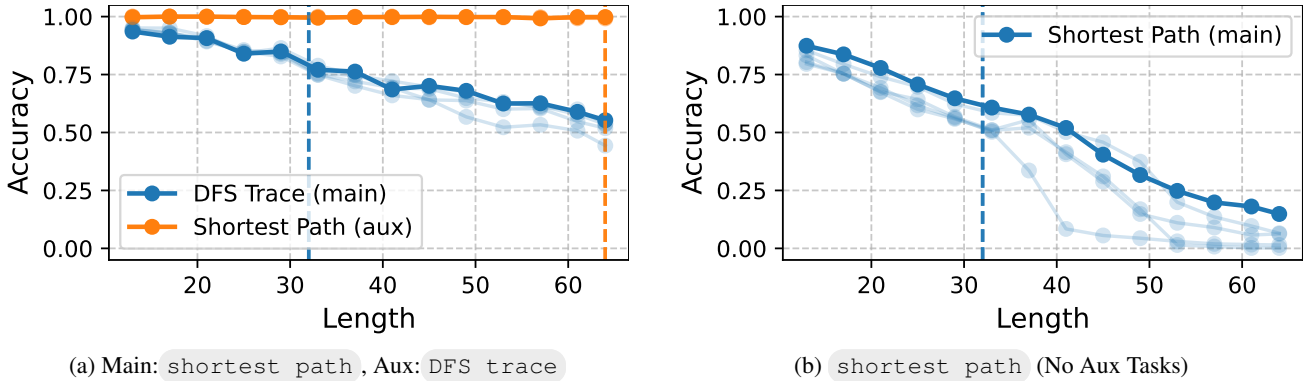

(a) Main: `shortest path`, Aux: `DFS trace`

(b) `shortest path` (No Aux Tasks)

Figure 7: Length generalization results for maze task group with reversed task roles. Each model is trained with 5 random seeds; best-performing runs are shown in bold lines. Co-training `shortest path` with `DFS trace` (a) leads to improved generalization over training on `shortest path` alone (b).

C.3. Length Generalization Transfer from Pretraining

Remarkably, we find that natural language pretraining can serve as an effective form of *implicit auxiliary task* that enhances length generalization in synthetic tasks. To explore this, we finetune various checkpoints of SmolLM-360M (Allal et al., 2024) on `addition` and `shortest path` tasks. SmolLM is released by Huggingface and pretrained on a diverse corpus containing natural language and programming data, which includes long-range structures and dependencies.

Before finetuning, we verify that the model does not already solve these tasks. For `addition`, a zero-shot evaluation using prompt-based input results in near-zero accuracy, confirming that the model has not learned this task during pretraining. For the maze task, all node tokens are newly introduced during finetuning, meaning the entire input format is unseen by the pretrained model.

We then finetune models from multiple publicly available checkpoints, taken throughout the pretraining process (from step 160K to 2.56M), and evaluate their length generalization performance on out-of-distribution inputs. As shown in Figure 10, we observe a clear trend: generalization to longer inputs improves steadily with pretraining progress, for both arithmetic and maze-solving tasks. This suggests that natural language pretraining instills reusable inductive biases that transfer to novel tasks—even when those tasks have little structural resemblance to natural language. We speculate the extent of generalization transfer from pretrained models may not be limited to length generalization, but could extend to other forms of out-of-distribution generalization such as compositional reasoning, distributional shifts, and task complexity. Future work could explore whether similar transfer effects exist for other generalization challenges.
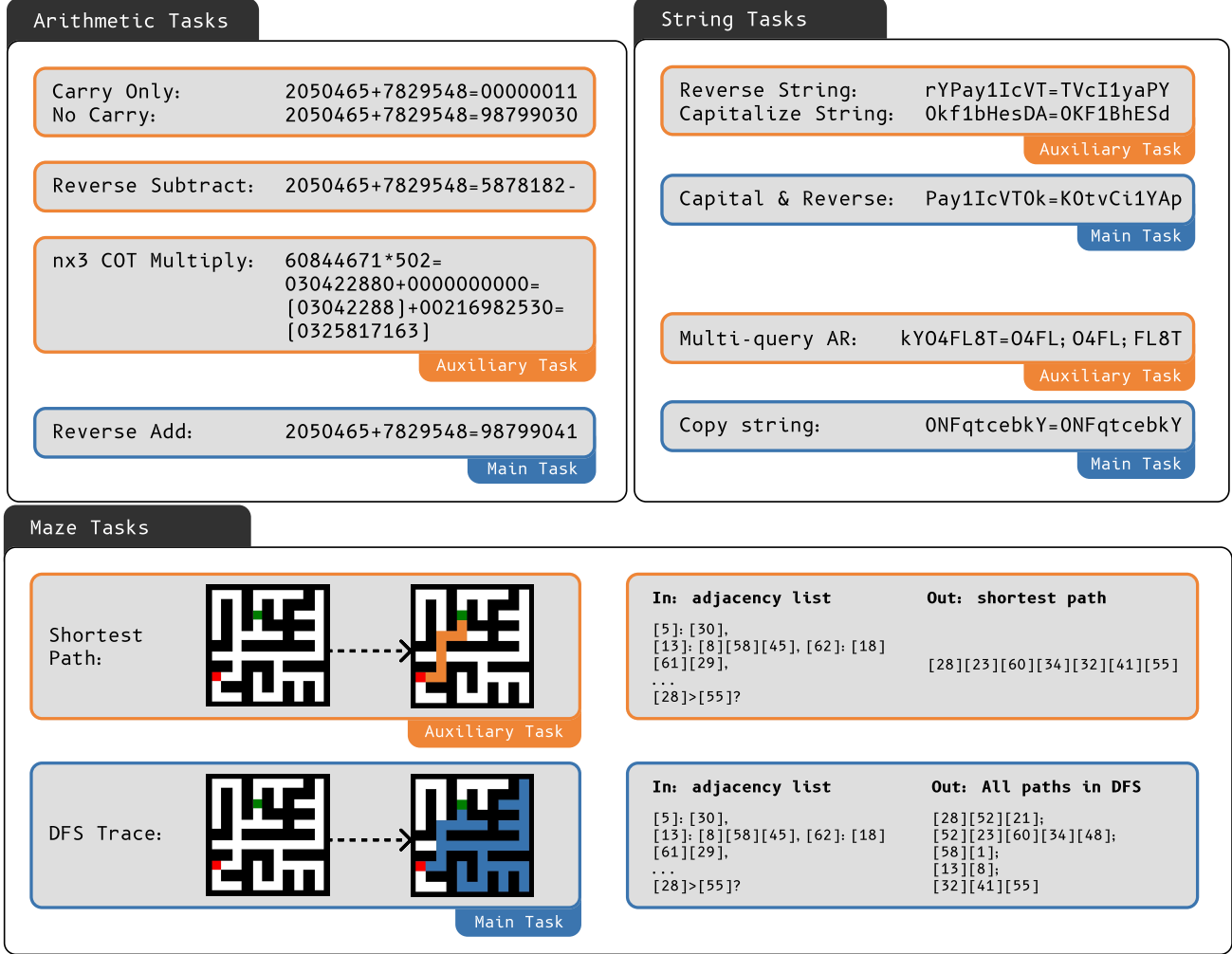
Figure 8: Overview of the tasks used in our length generalization transfer experiments, spanning three domains: arithmetic, string manipulation, and maze solving. Each group consists of a main task trained on shorter sequences and one or more auxiliary tasks trained on longer ones. We study whether generalization to longer inputs can be transferred from the auxiliary to the main task.

Additionally, we confirm that length generalization transfer is not limited to small models trained from scratch, but also emerges in finetuned pretrained models. Additional results across other task groups are provided in Appendix C.6.

C.4. Unstable Training Dynamics in Length Generalization Transfer

As shown in Figures 2, 3, and 6, not all random seeds exhibit successful length generalization transfer. In our experiments with 5 different seeds per task group, we observe considerable variability in length generalization transfer performance. The variability is entirely due to different model initializations, since we keep the dataset the same between runs. To better illustrate this instability, we visualize training dynamics in Figure 11.

The plots show training curves for the `addition` main task when co-trained with `no carry` and `carry only` auxiliary tasks. During evaluation, we sweep over input lengths from 1 to 36, which is classified into three regimes:

- In-distribution (length 1–16): These inputs fall within the training range for the main task. Accuracy in this regime improves quickly and remains stable.

- Expected transfer range (length 17–32): These inputs are unseen by the main task but seen by the auxiliary tasks.

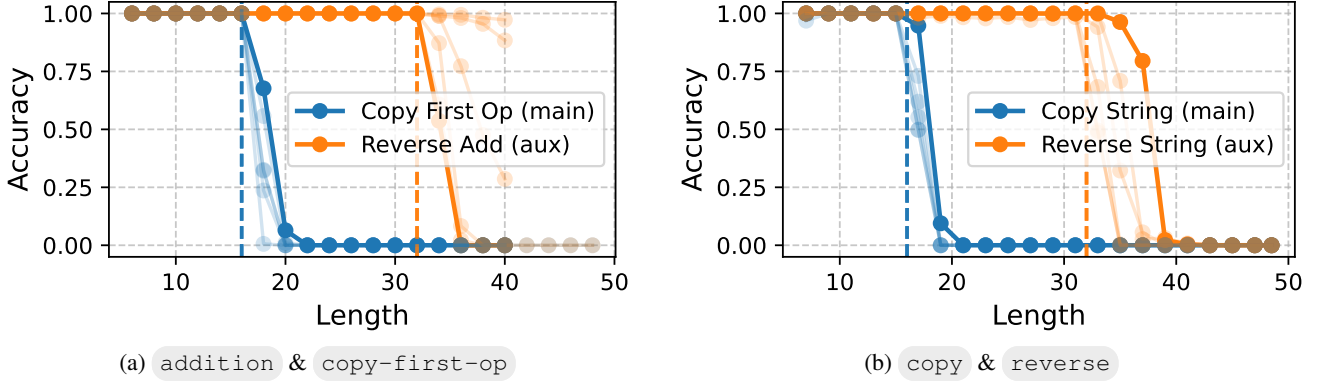(a) `addition` & `copy-first-op` (b) `copy` & `reverse`

Figure 9: Control tasks for (a) addition and (b) string operations. These unrelated task pairs fail to produce length generalization transfer, confirming that task relatedness is crucial.
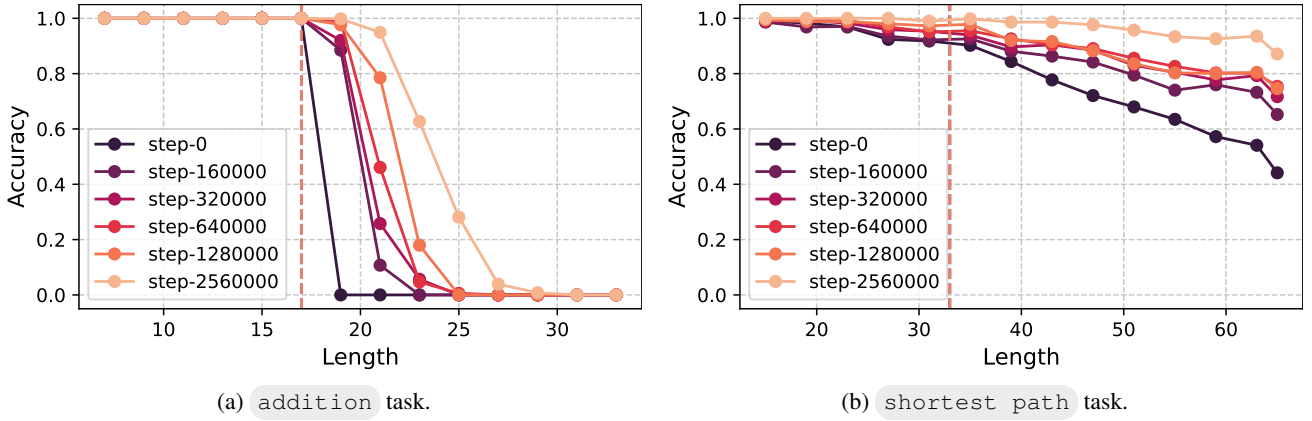

(a) `addition` task. (b) `shortest path` task.

Figure 10: Finetuning at different SmolLM-360M checkpoints reveals that length generalization transfer improves with more natural language pretraining.

Performance in this range is highly variable and sensitive to training dynamics.

- Fully OOD (length >32): These inputs are unseen by both the main and auxiliary tasks. As expected, accuracy in this regime remains low.

C.5. Additional Results on Arithmetic and String Tasks

For task groups with two auxiliary tasks– `addition` with `no carry` and `carry only`, and `flipcase-reverse` with `flipcase` and `reverse` –we additionally evaluate the effect of training with only one of the auxiliary tasks. As shown in Figure 12, length generalization transfer performance consistently declines when only a single auxiliary task is used, compared to co-training with both. Notably, the choice of auxiliary task matters: models trained with the more relevant auxiliary (`no carry` or `reverse`) exhibit stronger generalization than those trained with less relevant ones (`carry only` or `flipcase`). These results reinforce the importance of task alignment for successful transfer.
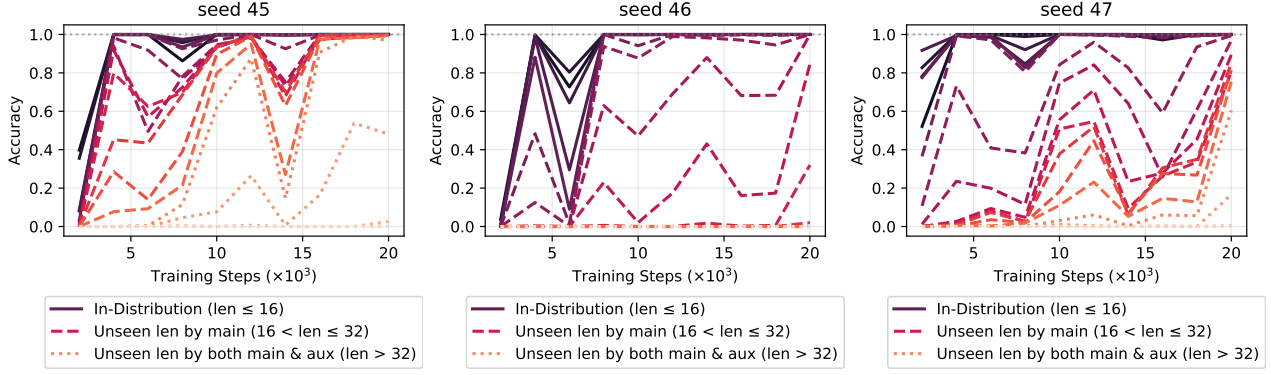
Figure 11: Training curves for the `addition` when co-trained with `no carry` and `carry only`. Accuracy in the transfer region (length 17–32) fluctuates significantly, illustrating unstable training dynamics in length generalization transfer.

Top row: A: `addition`, B: `no carry`, C: `carry only`

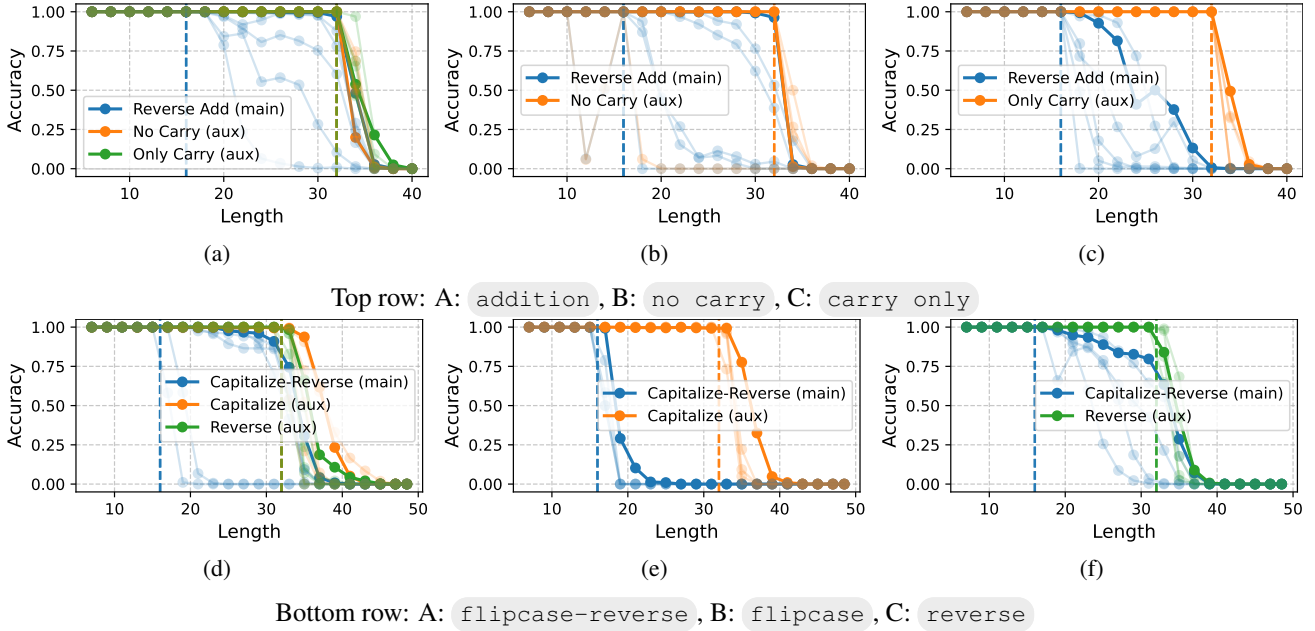Bottom row: A: `flipcase-reverse`, B: `flipcase`, C: `reverse`

Figure 12: Additional results for arithmetic and string (copy) task groups. Each row shows performance on the main task (A) when co-trained with: both auxiliary tasks (left), only one of the auxiliary task (middle & right). Performance degrades when training with only one auxiliary task, especially when the auxiliary is less structurally aligned with the main task.

C.6. Additional Pretrained Model Results

We replicate our length generalization transfer experiments using a pretrained language model, SmolLM-360M, where we observe similar patterns of length generalization transfer as in the from-scratch setting. Figure 13 presents results across three arithmetic task groups and one string manipulation group. As with our earlier experiments, co-training with structurally related auxiliary tasks facilitates generalization beyond the training length. Notably, we also confirm that control task pairs–such as `addition` with `copy-first-op`–do not lead to successful transfer. Orthogonal to the length generalization transfer, results show that SmolLM-360M exhibits strong inherent generalization in copying tasks (13c, 13d).

13

(a) A: `addition`, B: `subtraction`

(b) A: `addition`, B: `no carry`, C: `carry only`

(c) A: `addition`, B: `copy-first-op`

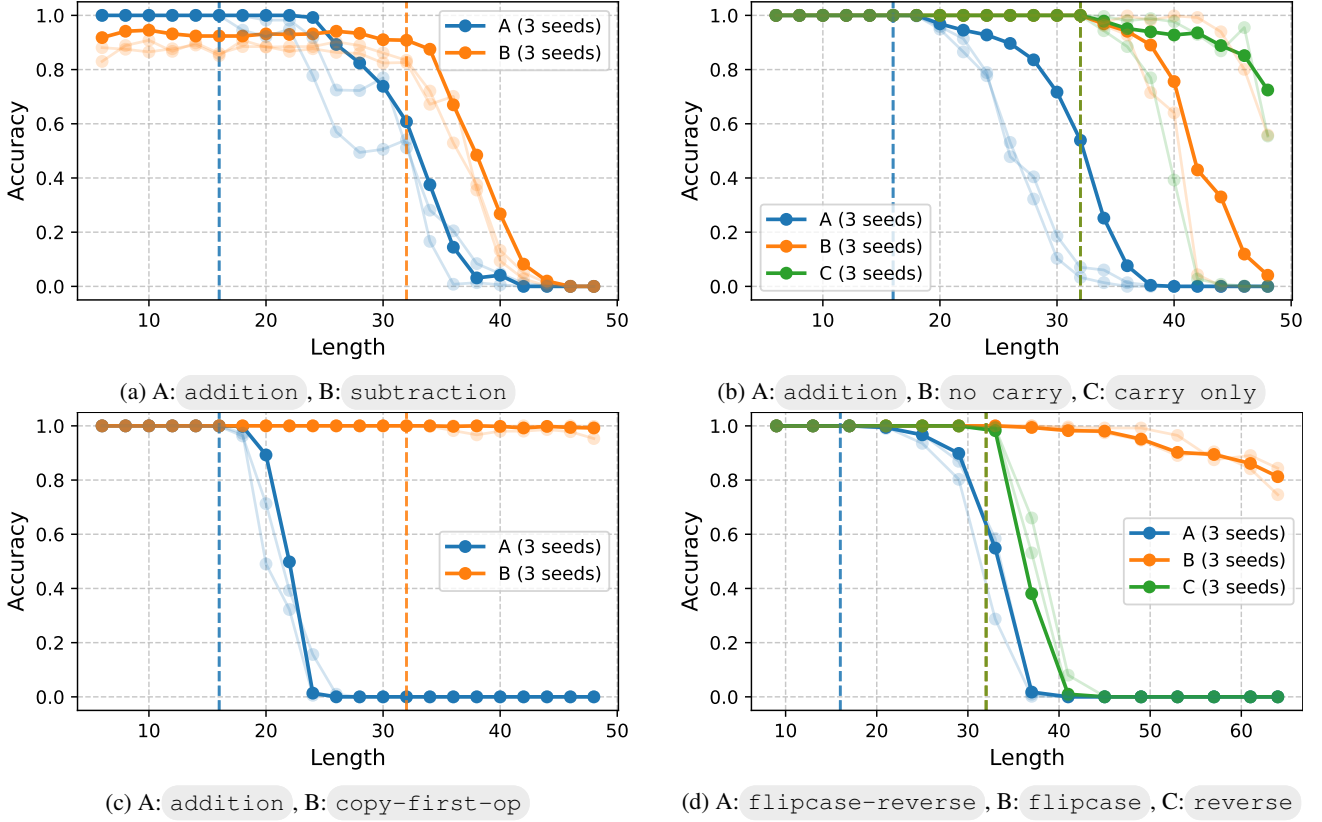(d) A: `flipcase-reverse`, B: `flipcase`, C: `reverse`

Figure 13: Length generalization transfer with the pretrained model SmolLM-360M. (a–c): Arithmetic task groups. In (a) and (b), we observe successful transfer from auxiliary to main tasks, mirroring results from from-scratch training. In (c), no transfer occurs when using the control task `copy-first-op`, confirming the importance of task relevance. (d): String manipulation task, showing transfer from `flipcase` and `reverse` to `flipcase-reverse`. Overall, the transfer effect persists in the pretrained model.

D. Ablations

D.1. Varying Main and Auxiliary Task Lengths

In our previous experiments, we fixed the main task length to 16 and the auxiliary task length to 32. A natural question is: does length generalization transfer persist across other main–auxiliary length configurations? To investigate this, we define the *generalization gap* (Figure 14), a scalar between 0 and 1 that quantifies the discrepancy in performance between the main and auxiliary tasks across a range of evaluation lengths. A smaller generalization gap indicates stronger transfer, with a value of 0 implying perfect alignment between the main and auxiliary generalization curves.

First we fix the task group `addition`, `no carry` and `carry only`. Then, we systematically vary the training lengths of both main and auxiliary tasks across the range $\{4, 8, 16, \ldots, 256\}$ and compute the average generalization gap over three random seeds. As shown in Figure 14, we find that the transfer effect is most effective when the ratio between the auxiliary and main lengths is between 0.5 and 2. The intuitive explanation is that, when the difference between task length is too high, the model will overfit to the task length difference and therefore do not exhibit length transfer.

D.2. Rotary Position Encoding Encourages Length Generalization Transfer

In length generalization literature, NoPE (No Positional Encoding) has often been favored for its length generalization performance on various tasks. However, in practice, many modern transformer models adopt RoPE (Rotary Positional Encoding), motivated by its strong empirical performance in long-context settings (Peng et al., 2023; Ding et al., 2024) and its ability to induce meaningful position-sensitive attention patterns (Barbero et al., 2024).
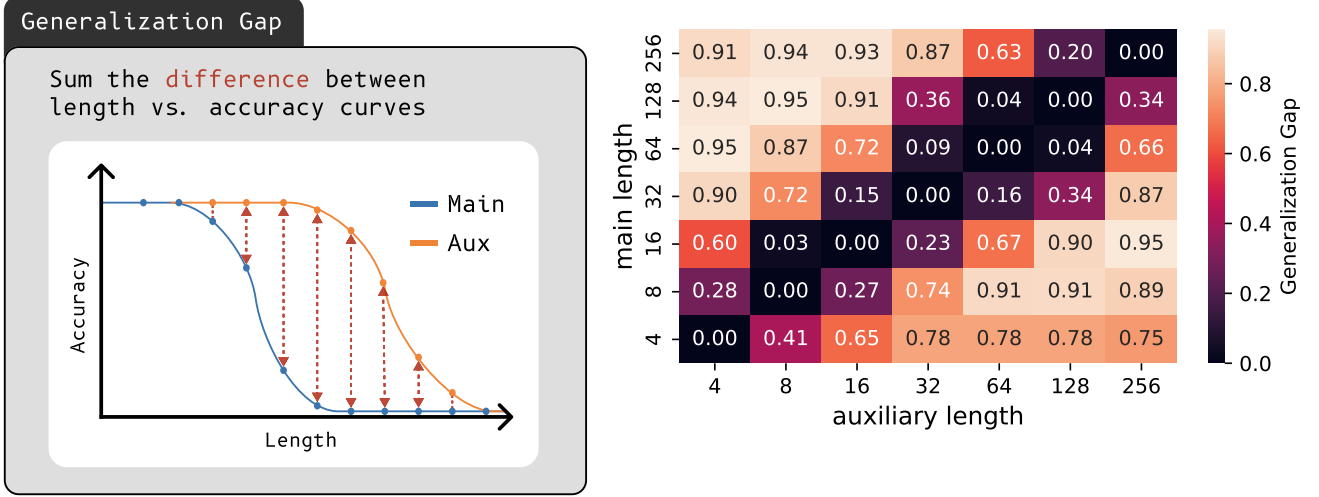
Figure 14: (a) The *generalization gap* is defined as the average difference in accuracy between the main and auxiliary tasks across evaluation lengths, normalized to the range [0, 1]. A lower value indicates better transfer. (b) Generalization gap across different combinations of main (`addition`) and auxiliary (`no carry` & `carry only`) training lengths. The transfer effect is strongest when the ratio between auxiliary and main lengths is between 0.5 and 2, as shown by the dark diagonal band.

To compare the two encoding strategies in the context of length generalization transfer, we re-run our main experiments using the same model architecture but remove the RoPE component. As shown in Figure 16, RoPE consistently outperforms NoPE in enabling generalization transfer from auxiliary to main tasks. This finding is orthogonal to the previous result (Kazemnejad et al., 2024) that NoPE is better suited for length generalization and potentially explains the superior performance of RoPE in real-world models and tasks.
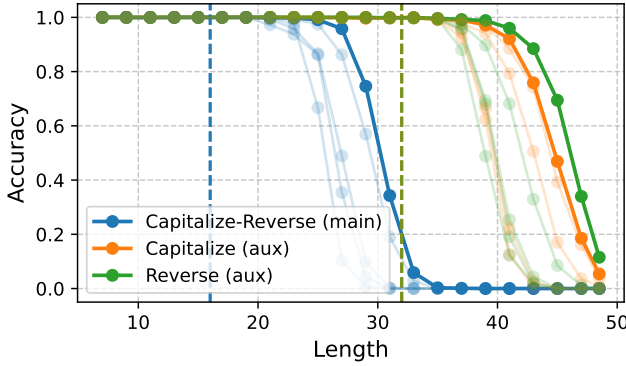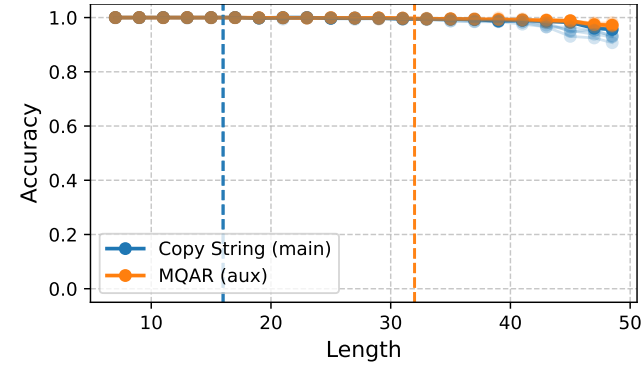
15
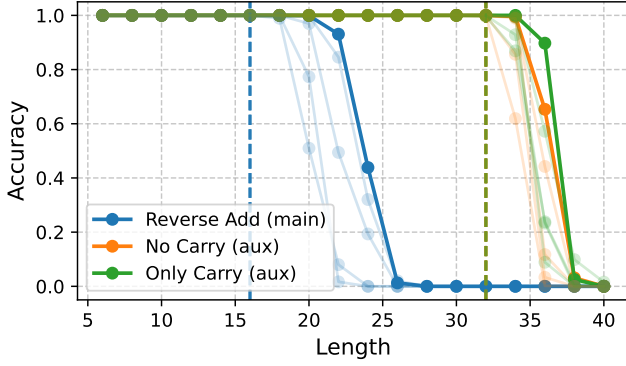
(a) Main: `addition`, Aux: `no carry` & `carry only`
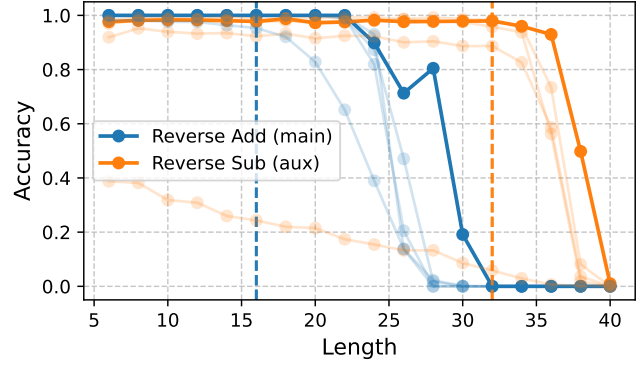
(b) Main: `addition`, Aux: `subtraction`

(c) Main: `flipcase-reverse`, Aux: `flipcase` & `reverse`

(d) Main: `copy`, Aux: `MQAR`

Figure 15: Length generalization transfer results using NoPE model, under the same task settings. The transfer effect is notably weaker in most tasks.
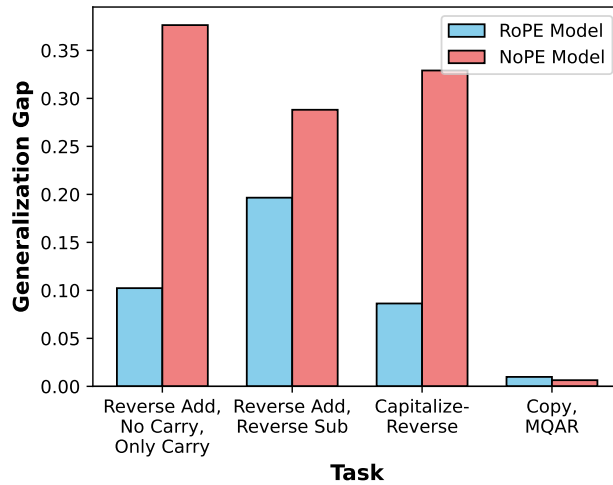

Figure 16: Comparison of generalization gap across several task groups shows that NoPE leads to significantly weaker transfer compared to RoPE.

16

D.3. Evidence of Circuit Sharing Between Tasks

In Section C.2, we established that mere exposure to longer inputs is insufficient for length generalization transfer–structural alignment between tasks is crucial. In this section, we present evidence that length generalization transfer coincides with the sharing of internal mechanisms between tasks. Specifically, we study whether transformer models reuse similar attention circuits across tasks when length generalization transfer occurs. We focus on two measures of similarity between task mechanisms:

- **Attention matrix difference**: the sum of entry-wise absolute differences between the attention matrices at each head between the two tasks.

- **Attention head mean-ablation map difference**: for each attention head, we measure the drop in task accuracy after replacing its output with the mean activation across the batch. This yields a 6×6 matrix (corresponding to 6 layers and 6 heads of the model) per task, representing the importance of each head. We then compute the average absolute difference between the two matrices to assess divergence in head importance.

The methodology used in our head ablation studies is known as *activation patching*. Prior works such as Wang et al. (2022); Cammarata et al. (2021); Olsson et al. (2022) have developed this technique to uncover the underlying computational circuits in language models.

Higher values for these metrics indicate more divergent computational mechanisms between tasks. We track how these metrics evolve over training and compare with the generalization gap (defined in Figure 14). Across different checkpoints that yield varying levels of generalization, we find that the attention similarity metrics correlate with the generalization gap. This suggests that when generalization improves, the internal attention patterns become more aligned across tasks, indicating increased sharing of computational mechanisms. The following analysis demonstrates this correlation across different task types.

D.3.1. EXAMPLE ATTENTION HEAD ABLATION MAPS

We first visualize the *attention-head mean-ablation maps* for a pair of related tasks— `addition` and `subtraction` —across four training checkpoints (Figure 17). Each 6×6 matrix represents the importance of each attention head: the value at position (i, j) indicates the drop in accuracy when head i in layer j is replaced with the mean activation across the batch. These matrices reflect the reliance of each task on specific attention heads. If two tasks reuse the same attention circuitry, their ablation maps should be similar. The scalar quantity used in subsequent comparisons is the average absolute difference between two such matrices.

(a) Task A, Ckpt 2000   (b) Task A, Ckpt 8000   (c) Task A, Ckpt 16000   (d) Task A, Ckpt 20000

(e) Task B, Ckpt 2000   (f) Task B, Ckpt 8000   (g) Task B, Ckpt 16000   (h) Task B, Ckpt 20000
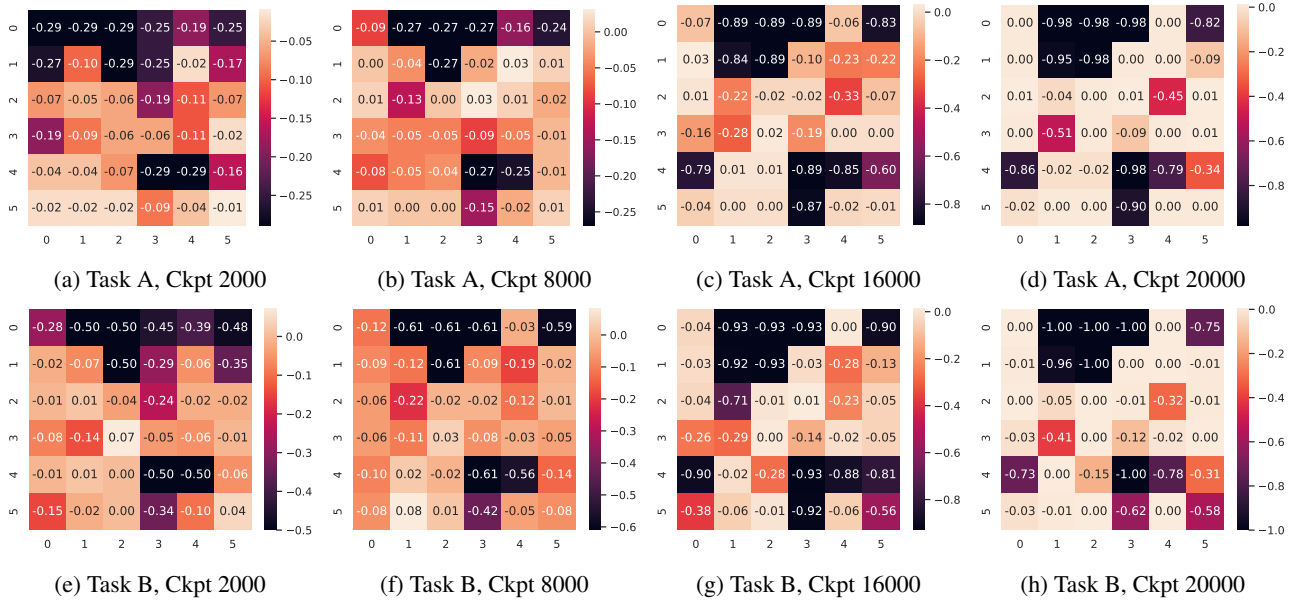
Figure 17: Mean-ablation maps for `addition` and `subtraction` across training checkpoints. Each (i, j) entry indicates the accuracy drop after mean-ablating head i in layer j. Similar ablation maps suggest that both tasks rely on overlapping computational circuits.

D.3.2. CORRELATION ANALYSIS ACROSS TASK TYPES

We now compare how two metrics (attention matrix difference and ablation map difference) track with the generalization gap across training checkpoints for different task categories.

String Tasks. As shown in Figure 18, the attention matrix difference does not correlate well with generalization in string task groups. However, the ablation map difference shows a clearer trend: as generalization improves, ablation similarity increases. This suggests that shared head usage, rather than raw attention weights, better reflects functional similarity across tasks.

(a) A: `flipcase-reverse`, B: `flipcase`, C: `reverse`

(b) A: `copy`, B: `MQAR`
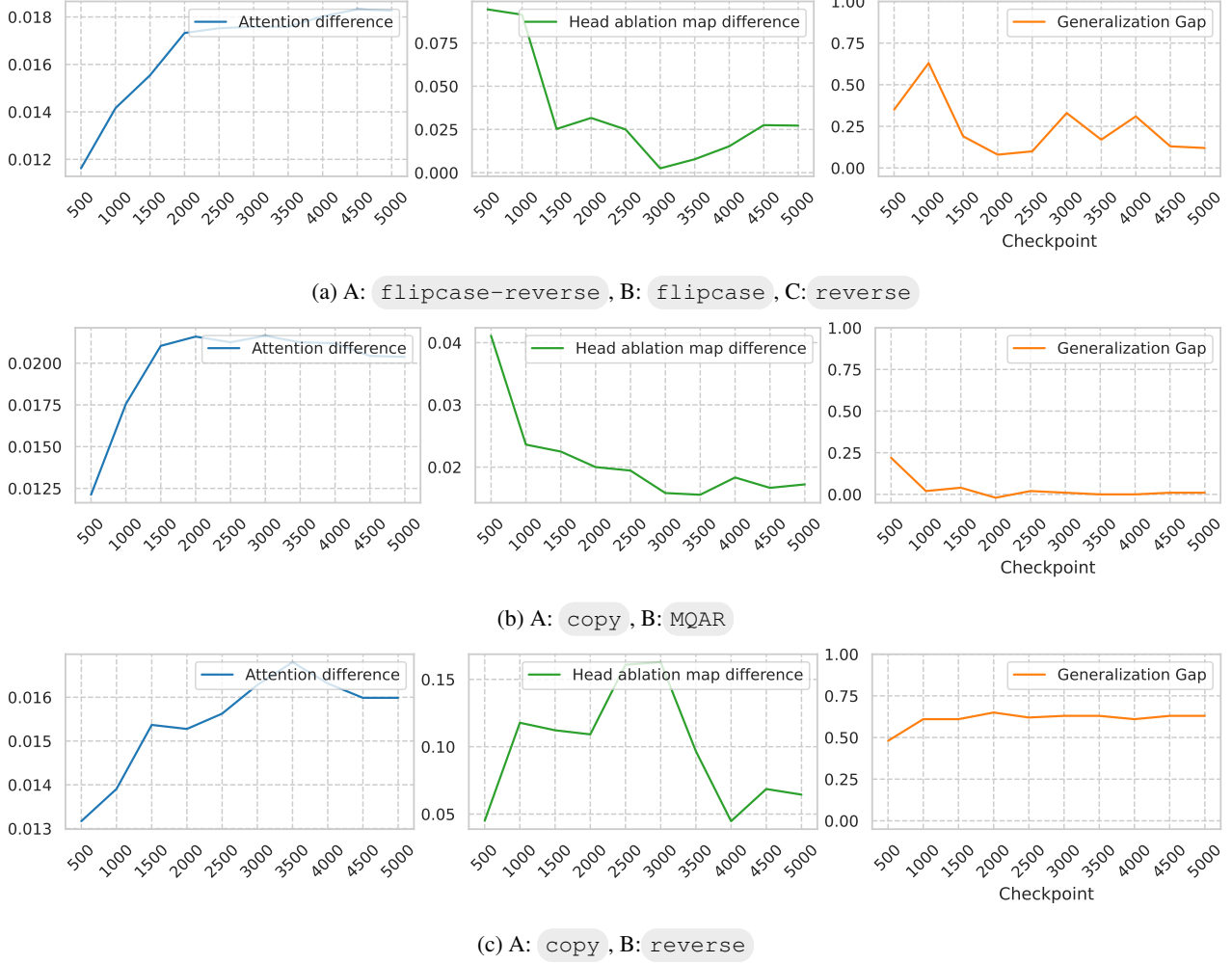

(c) A: `copy`, B: `reverse`

Figure 18: Circuit sharing results for string task pairs. The attention matrix difference does not correlate with generalization gap, while the head ablation map difference does, highlighting the relevance of shared attention head usage.

Arithmetic Tasks. In contrast, for arithmetic task pairs, both metrics strongly correlate with the generalization gap (Figure 19). This suggests that arithmetic tasks not only share similar head usage but also similar attention patterns at the matrix level.

Control Tasks. As a sanity check, we analyze a task pair with no expected transfer (`addition` and `copy-first-op`). As expected, neither metric correlates with generalization performance, reinforcing that our observed patterns are not incidental.

E. Experiment Details

E.1. Model

For all experiments, we use decoder-only transformer models following the Llama architecture. Unless otherwise specified, we use Rotary Positional Embeddings (RoPE) for positional encoding; exceptions are noted in the ablation studies in Section D.2.

For pretrained model experiments, we use SmolLM-360M (Allal et al., 2024), a compact transformer trained on natural language and code. Table 1 summarizes the model configurations used in our experiments.

(a) A: `addition`, B: `copy-first-op`

(b) A: `addition`, B: `no carry`

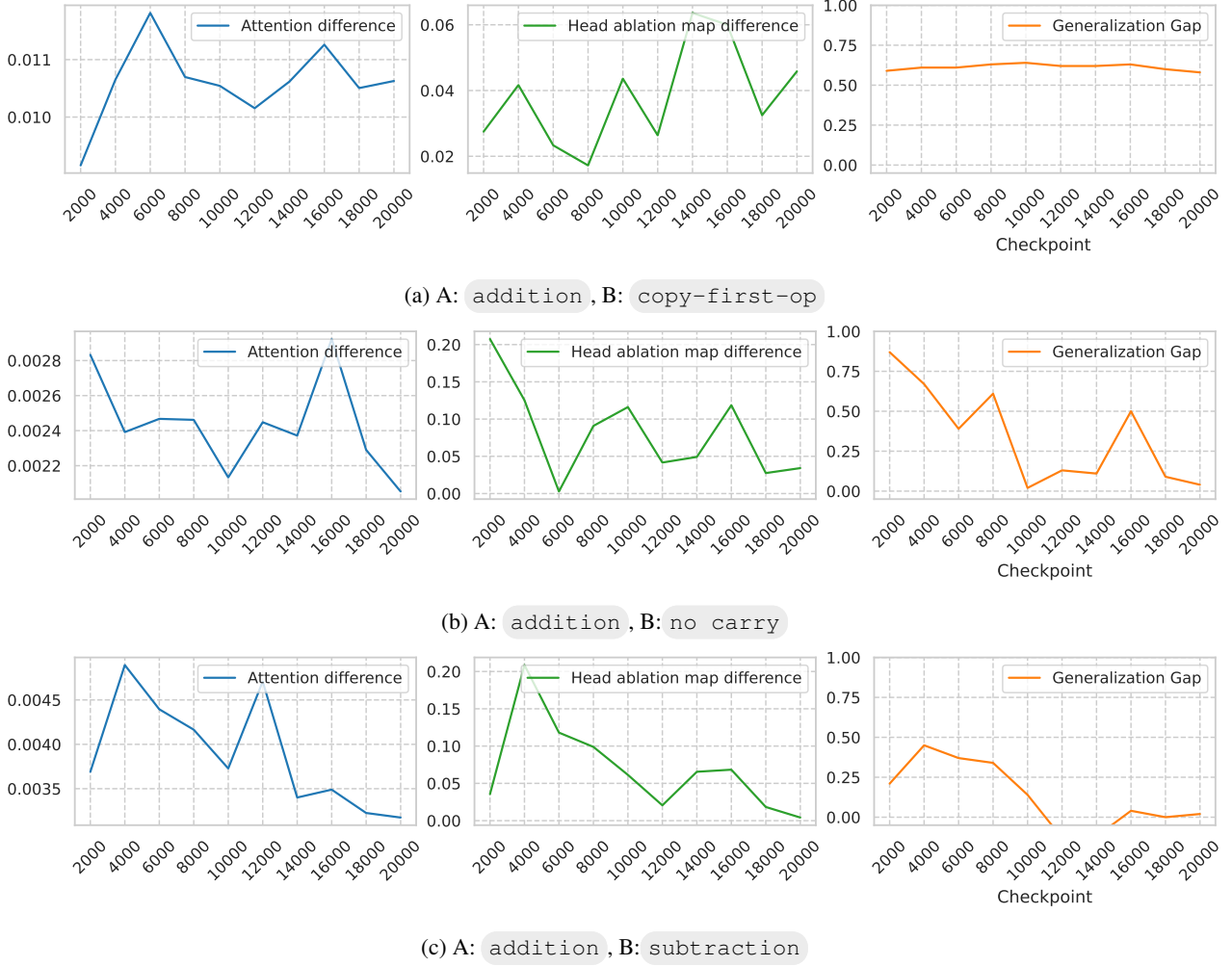(c) A: `addition`, B: `subtraction`

Figure 19: Circuit sharing results for arithmetic tasks. Both attention matrix and head ablation map differences correlate with generalization gap in related task pairs (b, c), but not in unrelated control task pairs (a).

Table 1: Model configurations used in our experiments.

Model	Self-Attn Layers	Num Heads	Embedding Dim
From-Scratch	6	6	384
SmolLM	32	15	2560

E.2. Data Formats and Data Sampling

We provide examples of each task in Table 2. For all arithmetic tasks, both the inputs and outputs are written in reverse digit order. For the `n × 3 CoT multiply` task, the output includes intermediate steps where the first operand is multiplied by each digit of the second operand.

For maze-based tasks, we serialize graphs using an adjacency list format with unique node tokens, followed by a query specifying the start and end node. A detailed example is shown in Figure 5.

20

Table 2: Examples of algorithmic tasks used in our experiments.

Task Name	Input	Output
`only carry`	82050465+23782955=	010010111
`no carry`	82050465+23782955=	057323100
`addition`	82050465+23782955=	067333211
`subtraction`	82050465+23782955=	692674000
$n \times 3$ `CoT multiply`	60844671*502=	030422880+0000000000= 03042288+00216982530= 0325817163
`copy`	fVOBA1fR=	fVOBA1fR
`Multi-Query Associative Recall`	fVOBA1fR=	fVOB;OBA1;
`reverse`	fVOBA1fR=	Rf1ABOVf
`flipcase`	fVOBA1fR=	Fvoba1Fr
`flipcase-reverse`	fVOBA1fR=	rF1abovF
`Shortest Path`	[0]:[10], [15]:[4][5], [11]:[1][3][5], [3]:[11], [4]:[2][15], [14]:[9][5], [10]:[0][9][13], [2]:[4], [1]:[11], [7]:[5], [13]:[8][10], [5]:[11][7][14][15], [12]:[8][6], [9]:[10][14], [8]:[12][13], [6]:[12] ?[12]>[2]?	[12][8][13] [10][9][14] [5][15][4][2]
`DFS trace`	[0]:[10], [15]:[4][5], [11]:[1][3][5], [3]:[11], [4]:[2][15], [14]:[9][5], [10]:[0][9][13], [2]:[4], [1]:[11], [7]:[5], [13]:[8][10], [5]:[11][7][14][15], [12]:[8][6], [9]:[10][14], [8]:[12][13], [6]:[12] ?[12]>[2]?	[12][6]; [12][8][13][10][9][14][5][11][1]; [11][3]; [5][15][4][2]

E.3. Experimental Settings

E.3.1. HYPERPARAMETER CONFIGURATIONS

Table 3 lists the hyperparameters used for training across different task domains and model types. From-scratch models are trained with a higher learning rate and larger batch sizes, while pretrained models (SmolLM-360M) use lower learning rates and shorter training schedules. All models are optimized using AdamW with a learning rate schedule that includes a warm-up phase, a constant phase, and a cosine decay phase.

Table 3: Hyperparameters for training

Task	Batch Size	LR	Iterations	Warmup Iter	Decay Iter
Arithmetic Tasks	1024	1e-3	20000	2000	5000
String Tasks	1024	1e-3	5000	500	1000
Maze Tasks	256	1e-3	20000	2000	5000
Arithmetic Tasks (SmolLM)	128	5e-5	2500	250	500
String Tasks (SmolLM)	128	5e-5	1000	100	500
Maze Tasks (SmolLM)	256	5e-5	2500	250	500

E.3.2. COMPUTATIONAL RESOURCES

For all experiments in the paper, we run on a single machine with two NVIDIA GeForce RTX 3090 graphics cards. For all experiment settings, each individual training run is at most 2 hours. The total estimate of compute used, in terms of hours on the 2-GPU machine, is around 300 hours.