Extrapolation by Association: Length Generalization Transfer in Transformers

Ziyang Cai[*]
University of Wisconsin-Madison

Nayoung Lee
University of Wisconsin-Madison

Avi Schwarzschild
Carnegie Mellon University

Samet Oymak
University of Michigan

Dimitris Papailiopoulos
University of Wisconsin-Madison
Microsoft Research

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].

[*]Corresponding author. zcai75@wisc.edu

Preprint. Under review.

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

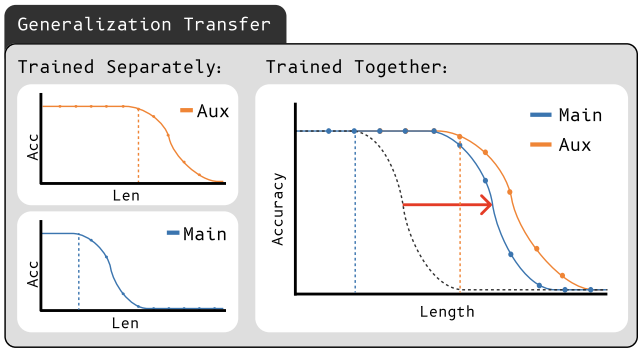


Figure 1: Trained separately, each task fails to generalize to longer inputs. When trained jointly, the main task inherits the generalization range of the auxiliary task.

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 replicates in pretrained language models, and that natural language pretraining transfers length generalization capabilities to synthetic downstream tasks.
3. We provide mechanistic evidence that transfer correlates with shared internal computation—specifically, the reuse of attention heads across tasks.

2 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., 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 reverse 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.

3 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, presented in reversed order.
  - `no carry` – Compute digit-wise sums mod 10, without carry propagation.
  - `carry only` – Output a binary mask indicating carry positions during addition.
  - `reverse subtract` – Compute the reversed digit-wise difference between two numbers.
  - $n \times 3$ `CoT multiply` – Multiply an n-digit number by 3, with chain-of-thought steps.
- **String Manipulation Tasks**
  - `string 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.
  - `capitalize` – Flip the case of all alphabetic characters (lower $\leftrightarrow$ upper).
  - `reverse` – Reverse the character order of the input string.
  - `capitalize-reverse` – Apply both reversal and case-flipping to the input string.
- **Maze Tasks**
  - `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.

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)

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 Section 4.3 for further details.

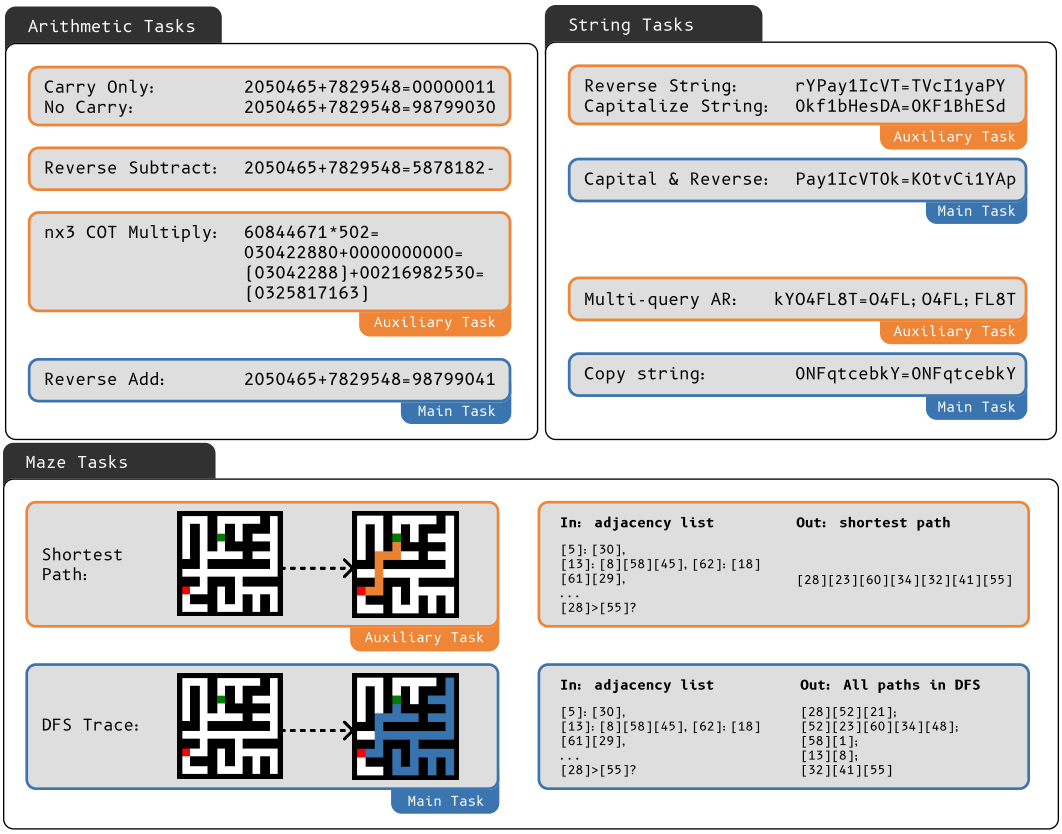


Figure 2: Overview of the tasks used in our length generalization transfer experiments, spanning three domains: arithmetic, string manipulation, and maze solving. Each group consists of a main task trained on shorter sequences and one or more auxiliary tasks trained on longer ones. We study whether generalization to longer inputs can be transferred from the auxiliary to the main task.

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 C.

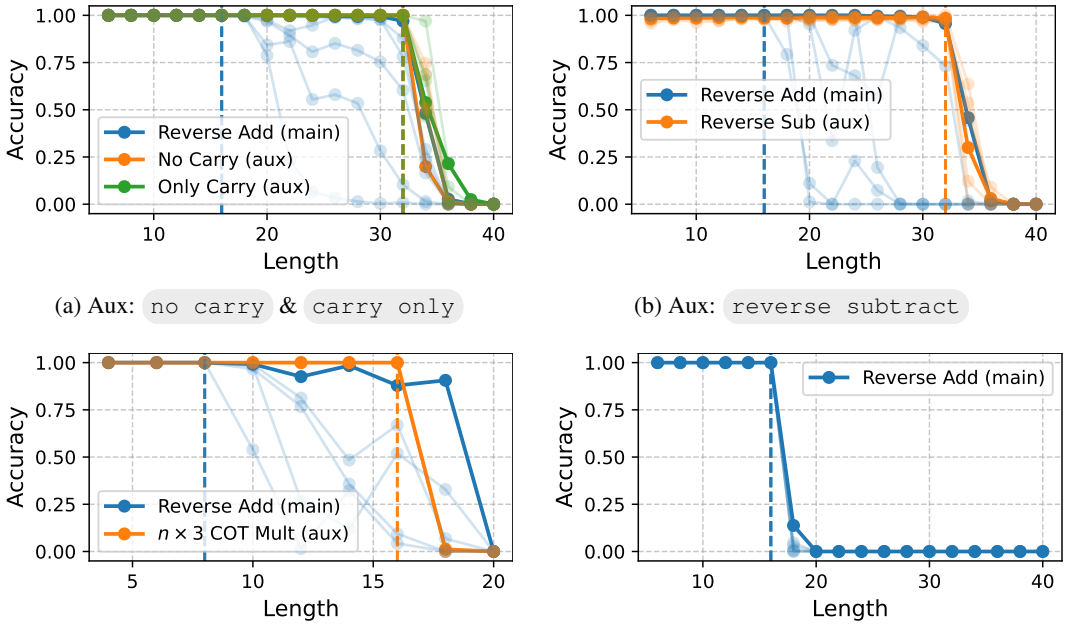
4 Length Generalization Transfer in Algorithmic Tasks

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. Figure 2 illustrates the three categories of tasks we study—arithmetic operations, string transformations, and maze navigation.

4.1 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.



(a) Aux: `no carry` & `carry only` (b) Aux: `reverse subtract`

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

Figure 3: 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.

As shown in Figure 3, models trained only on `reverse add` (Figure 3d) struggle to generalize beyond the training length. However, when co-trained with longer auxiliary tasks (Figures 3a, 3b, 3c), 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 Section 6.2.

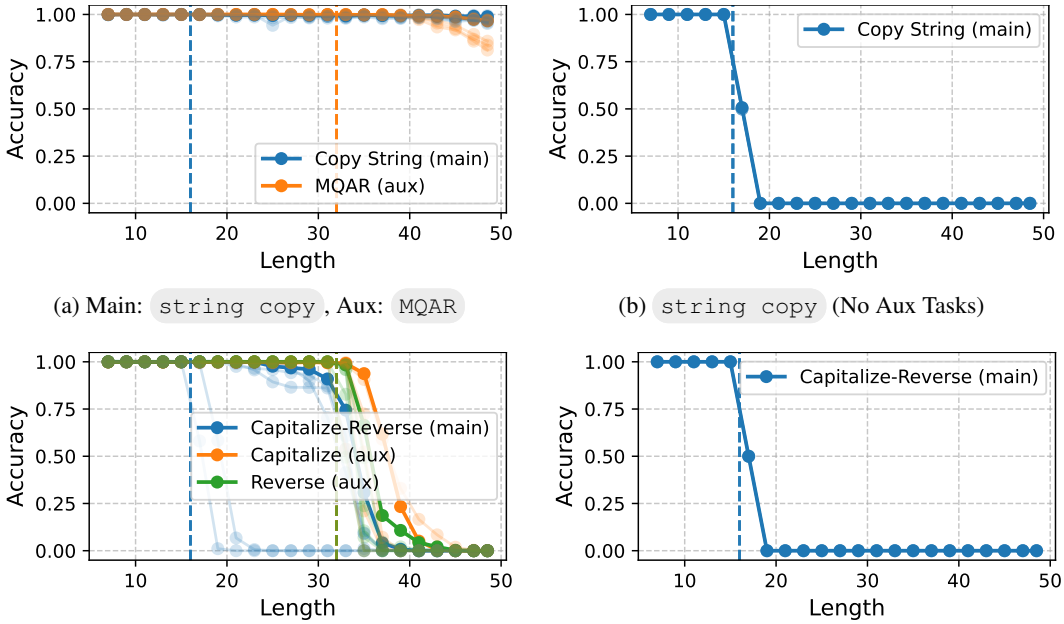
4.2 String Tasks

We now turn to string operations, where we observe similar transfer effects on two 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 4 shows that when trained on main tasks alone (Figures 4b, 4d), the model does not generalize beyond the training range. On the other hand,Adding training with auxiliary tasks enables substantial extrapolation (as shown in Figures 4a and 4c).

4.3 Maze Tasks

Lastly, we examine maze-solving tasks as a testbed for length generalization transfer. We define a maze as a spanning tree over a square grid, generated using Wilson's algorithm [Wilson, 1996], which



(a) Main: `string copy`, Aux: `MQAR` (b) `string copy` (No Aux Tasks)

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

Figure 4: 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.

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. Mazes are represented as adjacency lists, with each node and its neighbors encoded as individual tokens (e.g., [1], [2], ..., [64]). Input/output formatting examples are shown in Figure 2 and Section C.2.

A challenge in defining length generalization for mazes is that increasing grid size introduces unseen node tokens at test time. To avoid this, we fix the grid size and instead vary the number of nodes included in the spanning tree. Specifically, we define the input length as the total number of nodes in the maze graph and generate partial mazes by stopping Wilson's algorithm early. For example, to construct a 32-node maze on an 8×8 grid, we run the algorithm until 32 nodes are added. The resulting maze may not span the full grid but remains a valid traversal problem. Figure 5 illustrates such partial mazes with 16, 32, and 64 nodes.

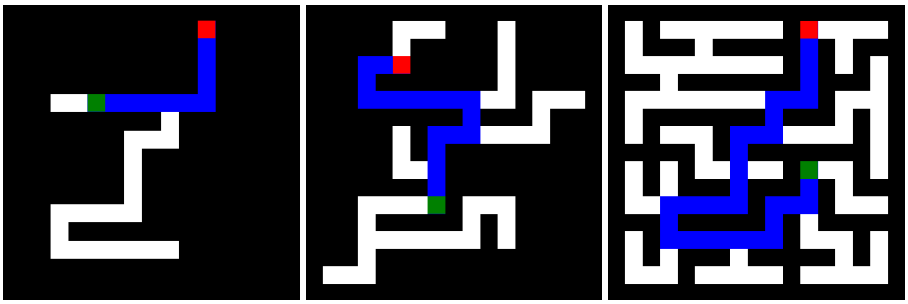
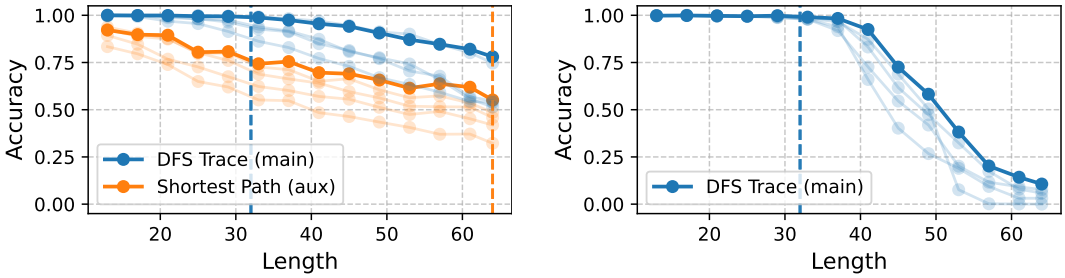


Figure 5: 8×8 mazes with number of nodes equal to 16, 32, and 64. We define length generalization as the ability to generalize to mazes with a higher number of nodes.

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. 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, which is shown in Figure 7.

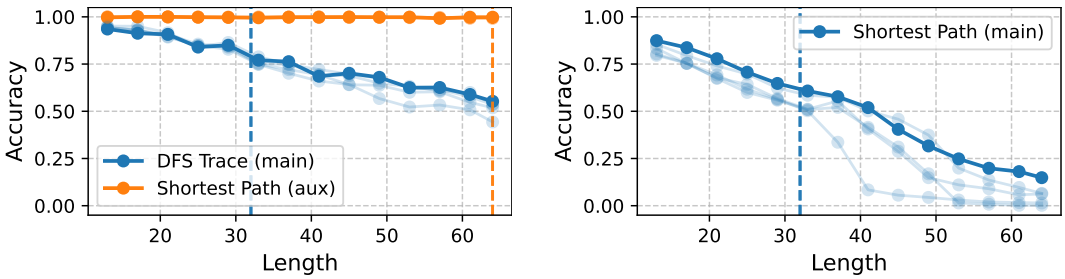


(a) Main: `DFS trace`, Aux: `shortest path` (b) `DFS trace` (No Aux Tasks)

Figure 6: Performance plots for maze tasks. Co-training `DFS trace` with `shortest path` (a) enables generalization to longer lengths compared to training on `DFS trace` alone (b).

4.3.1 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 4.3. 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.



(a) Main: `shortest path`, Aux: `DFS trace` (b) `shortest path` (No Aux Tasks)

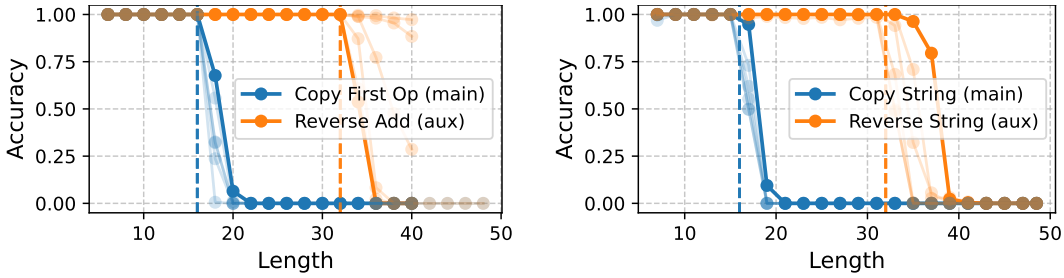
Figure 7: Length generalization results for maze task group with reversed task roles. Co-training `shortest path` with `DFS trace` (a) leads to improved generalization over training on `shortest path` alone (b).

4.4 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. or arithmetic, we use `copy-first-op`, which follows the addition format but simply copies the first operand. For string operations, we pair `string copy` with `reverse`. As expected, length generalization transfer is not observed with unrelated task (Figure 8).

5 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 `reverse add` and `shortest path`



(a) `reverse add` & `copy-first-op` (b) `string copy` & `reverse`

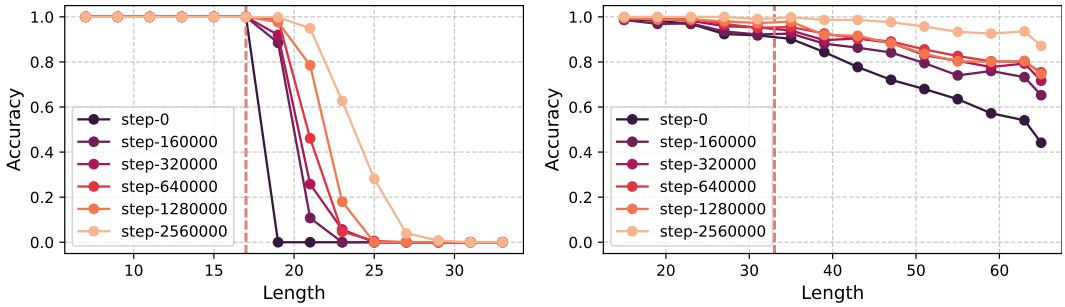
Figure 8: Control tasks for (a) addition and (b) string operations. These unrelated task pairs fail to produce length generalization transfer, confirming that task relatedness is crucial.

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 `reverse add`, 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 9, 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.

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 A.3.



(a) `reverse add` task. (b) `shortest path` task.

Figure 9: Finetuning at different SmolLM-360M checkpoints reveals that length generalization transfer improves with more natural language pretraining.

6 Ablations

In this section, we present several complementary analyses to better understand the conditions under which transfer occurs. We examine the effect of varying the length configurations of the main and auxiliary tasks and also provide an initial mechanistic explanation of the transfer phenomenon based

on circuit sharing between tasks. Additional analyses, including the instability of training dynamics (Section 6.2) and the effect of positional encodings (Section 6.3) are included in the Appendix.

6.1 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 10), 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 `reverse add`, `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 10, 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.

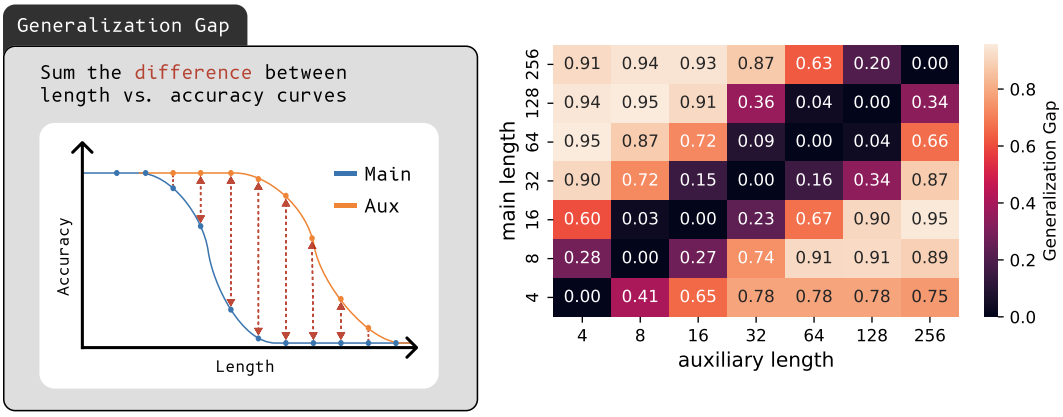


Figure 10: (a) The *generalization gap* is defined as the average difference in accuracy between the main and auxiliary tasks across evaluation lengths, normalized to the range [0, 1]. A lower value indicates better transfer. (b) Generalization gap across different combinations of main (`reverse add`) 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.

6.2 Unstable Training Dynamics in Length Generalization Transfer

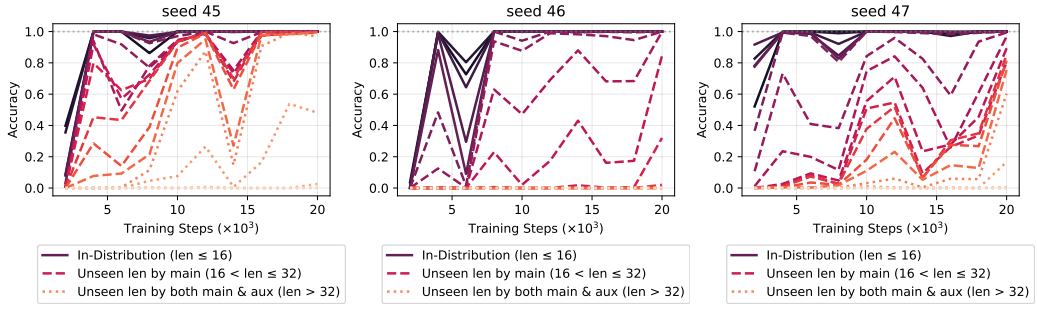


Figure 11: Training curves for the `reverse add` when co-trained with `no carry` and `carry only`. Accuracy in the transfer region (length 17–32) fluctuates significantly, illustrating unstable training dynamics in length generalization transfer.

As shown in Figures 3, 4, 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 `reverse add` 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. 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.

6.3 Rotary Position Encoding Encourages Length Generalization Transfer

In length generalization literature, *NoPE* (no positional encoding) is often favored for its strong extrapolation on individual tasks. However, many modern transformer models use *Rotary Positional Encoding* (RoPE) due to its empirical robustness in long-context and real-world settings [Peng et al., 2023, Ding et al., 2024, Barbero et al., 2024].

We re-evaluate our multitask transfer setup under both encoding schemes. Across task families, RoPE consistently yields stronger length-generalization *transfer* from auxiliary to main tasks. Detailed per-task curves are provided in Appendix A.1. Figure 12 summarizes the overall trend. This finding is orthogonal to the previous understanding that NoPE is better suited for length generalization and potentially explains the superior performance of RoPE in real-world models and tasks.

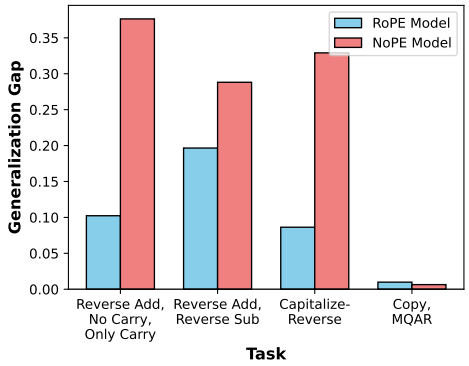


Figure 12: Comparison of generalization gap across task groups. Smaller gap means stronger transfer. RoPE consistently outperforms NoPE, indicating that rotary embeddings better support cross-task extrapolation.

7 Limitations

While our work demonstrates length generalization transfer across a range of synthetic tasks, several important limitations remain. First, our study does not provide a formal theoretical framework for understanding when and why transfer occurs. Without a principled understanding of the underlying mechanisms, predicting or optimizing transfer remains challenging. Second, our experiments are limited to relatively simple algorithmic domains with well-defined length parameters and deterministic solution paths. While this setup allows for controlled comparisons, it is unclear whether similar transfer effects would hold in settings that involve hierarchical reasoning, abstract problem-solving, or tasks requiring integration of multiple skills simultaneously. Addressing these limitations is a promising direction for future work and could further illuminate the generalization capabilities of transformer models in more realistic settings.

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.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions, including investigating length generalization transfer across related tasks, demonstrating this phenomenon across algorithmic tasks, observing similar effects in pretrained language models, and providing mechanistic evidence related to attention head reuse.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses limitations in Section 7, acknowledging the lack of a comprehensive theoretical framework, the focus on relatively simple algorithmic tasks, and open questions about how length generalization transfer might manifest in more complex scenarios.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper is primarily empirical rather than theoretical in nature, and doesn't present formal theorems requiring proofs.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 and Appendix B provide comprehensive details about experimental settings, including model architectures, task definitions, data sampling procedures, training parameters, and evaluation methods necessary to reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is included in the supplementary materials section of the submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all relevant training and test details in Section 3 and Appendix B, including model architectures, task specifications, data generation procedures, and hyperparameter configurations as shown in Tables 1-3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports results across 5 different random seeds for model initialization (mentioned in Section 3), and the figures show multiple runs with different random seeds, providing visual indication of result variability.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides type of compute, amount of compute used for each experiment run as well as estimate of total compute in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: Yes

   Justification: The authors have reviewed and verified that the paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The fundamental nature of the research on transformer models' length generalization capabilities suggests limited direct societal implications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper works with synthetic algorithmic tasks rather than models or datasets that pose risks of misuse, so safeguards aren't applicable.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper properly cites previous work and models used, including SmolLM and other transformer architectures, attributing them to their creators with appropriate citations.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper doesn't introduce new datasets or models intended as assets for the broader community.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This research doesn't involve human subjects or crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: No human subjects are involved in this research.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: While the paper studies transformer models, it doesn't use LLMs as part of its research methodology.

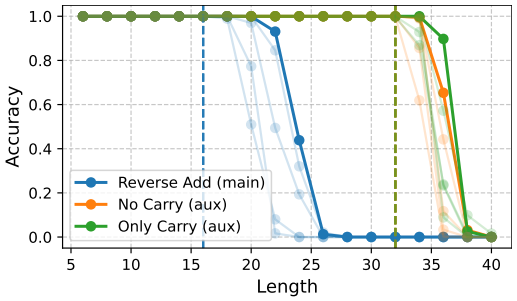
Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

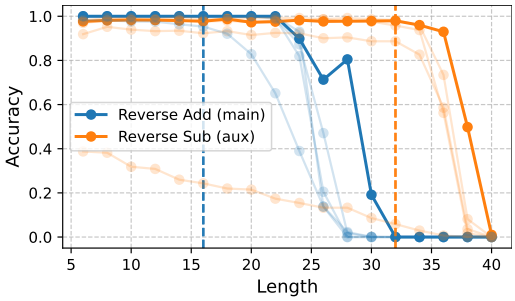
A Additional Results

A.1 Detailed Plots for Rotary vs. NoPE Models

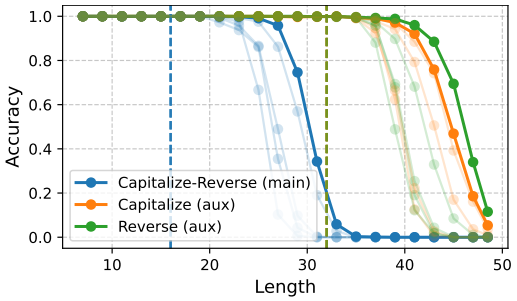
To complement Section 6.3, we show full length-generalization curves for the *No Positional Encoding* (NoPE) and *Rotary Positional Encoding* (RoPE) variants under the same task settings. Each subplot reports exact-match accuracy versus input length. The dashed vertical line indicates the maximum training length for each task. In all domains, RoPE models exhibit smoother extrapolation and better alignment between main and auxiliary tasks.



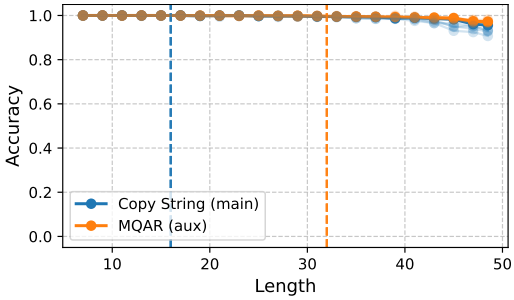
(a) Main: `reverse add`, Aux: `no carry` & `carry only`



(b) Main: `reverse add`, Aux: `reverse subtract`



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



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

Figure 13: Detailed performance curves comparing RoPE and NoPE variants across arithmetic and string tasks. RoPE models maintain strong transfer to longer lengths, while NoPE variants degrade rapidly beyond the training range.

A.2 Additional Results on Arithmetic and String Tasks

For task groups with two auxiliary tasks– `reverse add` with `no carry` and `carry only`, and `capitalize-reverse` with `capitalize` and `reverse` –we additionally evaluate the effect of training with only one of the auxiliary tasks. As shown in Figure 14, 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 `capitalize`). These results reinforce the importance of task alignment for successful transfer. As shown in Figure 14, 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 `capitalize`). These results reinforce the importance of task alignment for successful transfer.

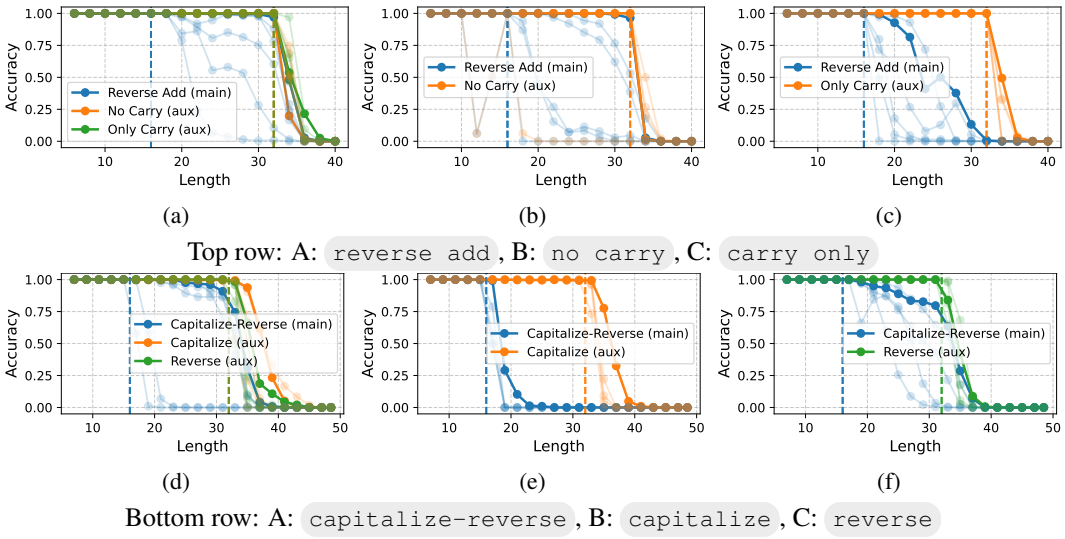
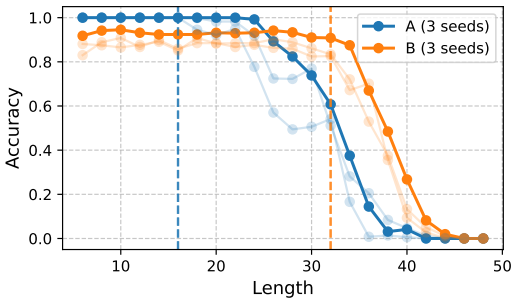


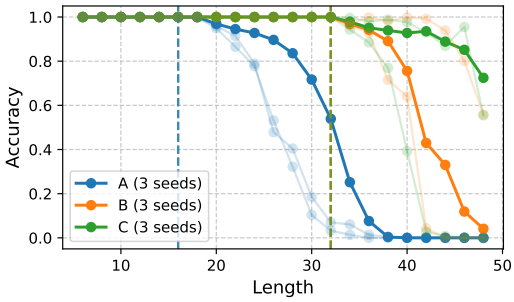
Figure 14: Additional results for arithmetic and string (copy) task groups. Each row shows performance on the main task (A) when co-trained with: both auxiliary tasks (left), only one of the auxiliary task (middle & right). Performance degrades when training with only one auxiliary task, especially when the auxiliary is less structurally aligned with the main task.

A.3 Finetuning from Pretrained Models

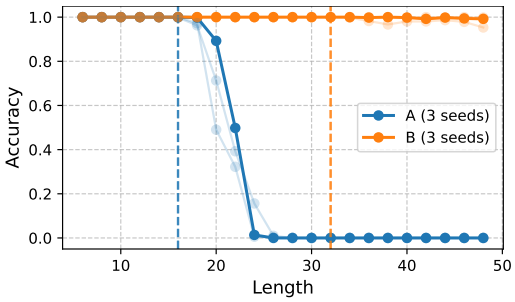
We replicate our length generalization transfer experiments using a pretrained language model, SmolLM-360M, where we observe similar patterns of length generalization transfer as in the from-scratch setting. Figure 15 presents results across three arithmetic task groups and one string manipulation group. 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 `reverse add` 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 (15c, 15d).



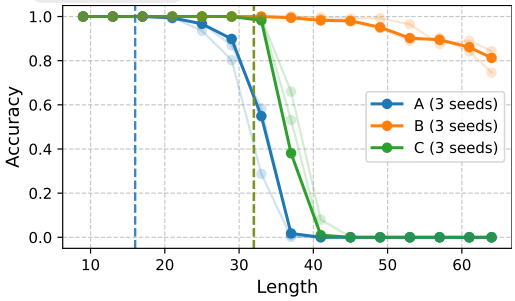
(a) A: `reverse add`, B: `reverse subtract`



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



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



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

Figure 15: 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 `capitalize` and `reverse` to `capitalize-reverse`. Overall, the transfer effect persists in the pretrained model.

B Mechanistic Evidence of Circuit Sharing

In this section we consolidate the mechanistic analysis of *length generalization transfer*, showing that successful transfer coincides with reuse of internal attention circuits across related tasks.

Metrics and protocol. We study whether transformer models reuse similar attention mechanisms across tasks when length generalization transfer occurs. We use two complementary metrics:

- **Attention matrix difference:** sum of entry-wise absolute differences between attention matrices (per head) for two tasks. Lower values indicate more similar attention patterns.
- **Attention-head mean-ablation map difference:** for each head (6 layers $\times$ 6 heads), we replace its output with the batch mean and measure the accuracy drop (activation patching). This produces a head-importance map per task; we then take the average absolute difference between the two maps. Lower values indicate more similar head usage.

This follows standard activation-patching methodology used in mechanistic-interpretability studies [Wang et al., 2022, Cammarata et al., 2021, Olsson et al., 2022]. Across checkpoints, reductions in the *generalization gap* (defined in Fig. 10) generally coincide with smaller differences in both metrics—i.e., tighter alignment of attention mechanisms across tasks when transfer strengthens.

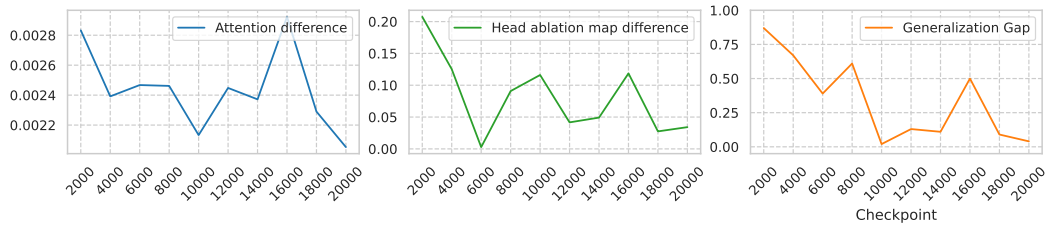


Figure 16: Arithmetic task group (`reverse add` with auxiliaries `reverse subtract` / `no carry`). Evolution of generalization gap, attention-matrix difference, and head mean-ablation map difference across checkpoints. When transfer improves (smaller gap), attention mechanisms align (smaller differences).

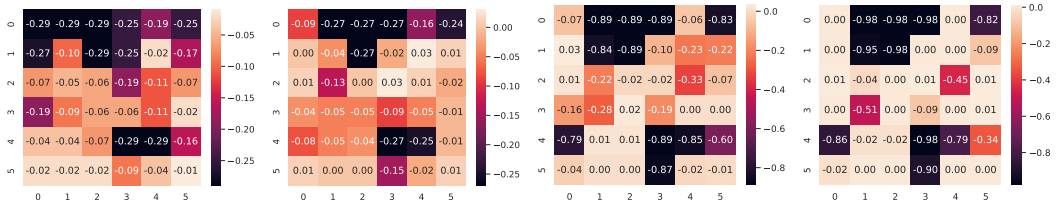
B.0.1 Example Attention-Head Ablation Maps

We visualize the *attention-head mean-ablation maps* for a pair of related tasks—`reverse add` and `reverse subtract`—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 reveal which heads are functionally critical for each task. If two tasks reuse the same circuitry, their ablation maps will appear similar; our scalar similarity metric is the average absolute difference between the two matrices.

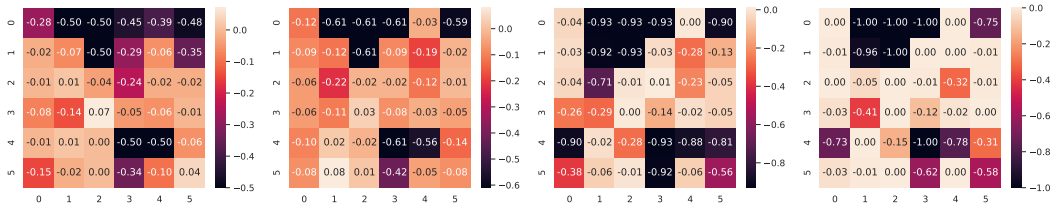
B.0.2 Extended Results Across Tasks

We next compare how the two circuit-similarity metrics track with the generalization gap across training checkpoints for string, arithmetic, and control tasks.

String tasks. Figure 18 shows that the raw attention-matrix difference does not correlate well with generalization for string tasks, whereas the ablation-map difference does. Shared head usage, rather than raw attention weights, better captures functional similarity.

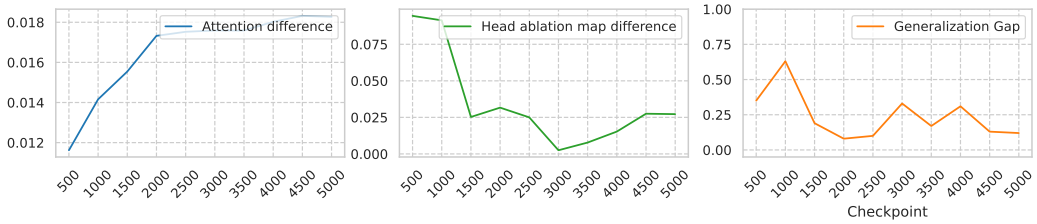


(a) Task A, 2k steps (b) Task A, 8k steps (c) Task A, 16k steps (d) Task A, 20k steps

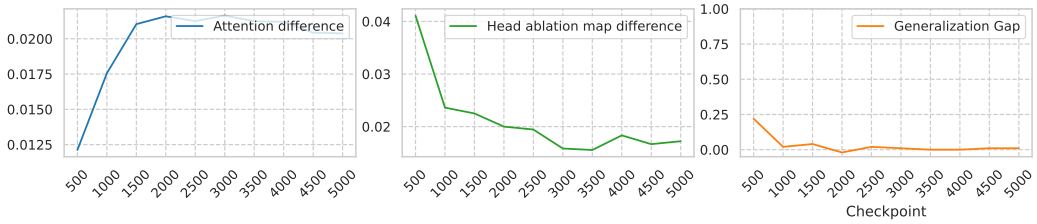


(e) Task B, 2k steps (f) Task B, 8k steps (g) Task B, 16k steps (h) Task B, 20k steps

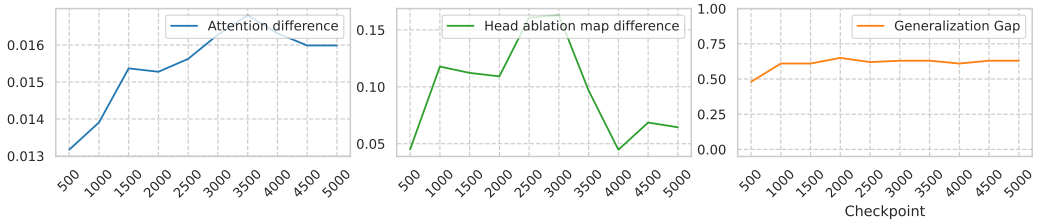
Figure 17: Mean-ablation maps for `reverse add` and `reverse subtract` across checkpoints. Each (i, j) entry shows the accuracy drop after mean-ablating head i in layer j. Similar maps indicate overlapping computational circuits.



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



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

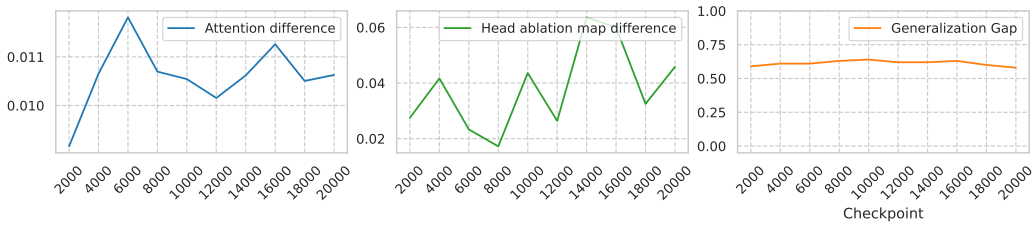


(c) A: `copy`, B: `reverse`

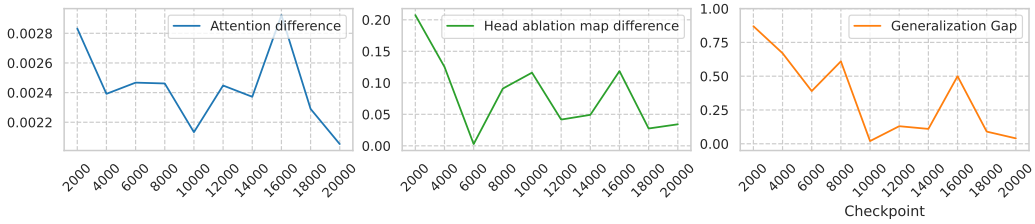
Figure 18: Circuit-sharing results for string task pairs. The attention-matrix difference shows weak correlation with generalization, whereas the head-ablation map difference tracks it closely, highlighting shared head usage.

Arithmetic tasks. For arithmetic task pairs (Figure 19), both metrics strongly correlate with the generalization gap, suggesting that these tasks share not only head usage but also detailed attention-pattern structure.

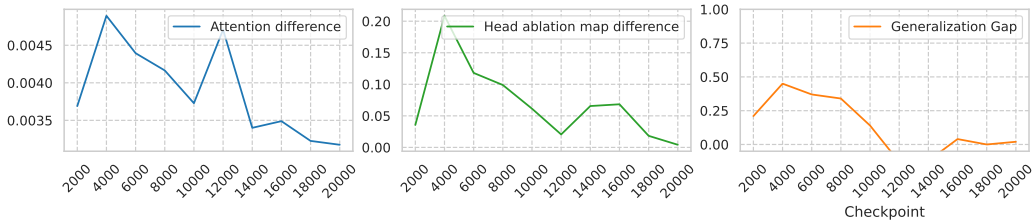
Control tasks. For unrelated task pairs such as `reverse add` with `copy-first-op`, neither metric correlates with performance, confirming that the observed correlations are not incidental.



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



(b) A: `reverse add`, B: `no carry`



(c) A: `reverse add`, B: `reverse subtract`

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 the unrelated control pair (a).

Takeaway. Across arithmetic, string, and maze domains (not shown), strong length-generalization transfer coincides with shared attention-head usage between tasks. This supports the hypothesis that transformers reuse a compositional "scaffold" of attention circuits when transferring extrapolation behavior across related problems.

C Experiment Details

C.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 6.3.

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.

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

C.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 `$n \times 3$ 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 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
`reverse add`	82050465+23782955=	067333211
`reverse subtract`	82050465+23782955=	692674000
`$n \times 3$ CoT multiply`	60844671*502=	030422880+0000000000= 03042288+00216982530= 0325817163
`copy string`	fVOBA1fR=	fVOBA1fR
`Multi-Query Associative Recall`	fVOBA1fR=	fVOB;OBA1;
`string reverse`	fVOBA1fR=	Rf1ABOVf
`capitalize`	fVOBA1fR=	Fvoba1Fr
`capitalize-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]

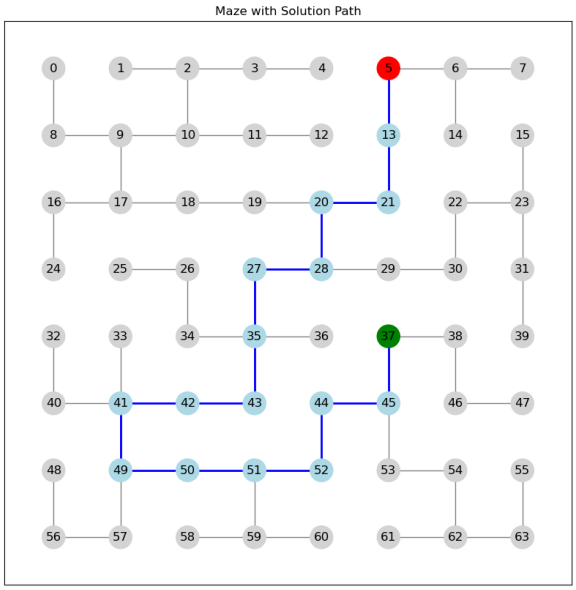


Figure 20: Detailed example of maze data format. Each node is a random number selected from $n \times n$ nodes in the grid.

C.3 Experimental Settings

C.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

C.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.